EVALUATING THE DYNAMICS OF AGENT-ENVIRONMENT INTERACTION

by

Dani Goldberg

A Dissertation Presented to the

FACULTY OF THE SCHOOL OF ENGINEERING

UNIVERSITY OF SOUTHERN CALIFORNIA

In Partial Fulfillment of the

Requirements for the Degree

DOCTOR OF PHILOSOPHY

(Computer Science)

May 2001

Dedication

This dissertation is dedicated to
the memory of my grandfather
Mirza
who died in May of 1998
during my second year as a Ph.D. student.

Acknowledgements

It is well-known that dissertations are seldom read, and I do not realistically expect this dissertation to be any different. But, in the unlikely event that someone does open these pages, I hope that he or she spends enough time on these words to realize that there are many people who have been important to this work and deserve sincere thanks.

Sincerest thanks first to my advisor, Maja Matarić, for the countless things (both big and small) that have helped make this dissertation and the completion of my Ph.D. studies a reality. Maja's energy, insight, humor, and many talents are an inspiration. Instead of attempting to thank her for everything (assuming that is even possible), I will simply mention, as an example, this: that anytime I went to her office uncertain, depressed or indifferent about the state of my research, she always had advice that revitalized my enthusiasm. How do you thank someone for such a rare talent?

Heartfelt thanks to the other four members of my dissertation defense committee: to Stefan Schaal, for his uncanny ability to concretize difficult concepts and his advice on comparing parametric and nonparametric AMMs; to Gaurav Sukhatme for detailed draft comments and much appreciated advice on handling the bureaucracy of graduating; to Kurt Palmer, for excellent feedback that has helped clarify the use of parametric and nonparametric statistics in the dissertation; and to Deborah Estrin, for keen questions and comments that have honed my presentation of the work. Many thanks also to Leslie Pack Kaelbling, whose comments on my dissertation proposal have helped me clarify the presentation of AMMs and their relationship to other models. Thanks also to Milind Tambe for great comments on several of my earliest AMM-related talks. Very special thanks also to Sridhar Mahadevan for help in establishing the connection between AMMs and SMPs.

Thanks to Barry Werger, my oldest labmate and compatriot, for friendship, late night singing sessions, sleepovers in the Lab, calzones on Fridays, movie nights, racquetball, sword-fighting stress-relief, being the only other Macintosh person in the Lab, and many other things, both personal and academic. The past five years would have been much more difficult without Barry's support. I would also like to thank my other labmates for making the Interaction Lab such a great place to work, and just be. Thanks to Monica Nicolescu for always being willing to lend a hand, from reading a paragraph that doesn't sound right, to setting up the video camera for my defense. Her positive outlook is contagious, but alas, if my desk is any indication, it seems that her neatness is not. Thanks to Brian Gerkey for great coffee, great humor, and guru-like advice on everything Unix. Few people are so patient and unselfish with their help. Thanks to Chad Jenkins for giving the lab a deep sense of camaraderie and family, and for the ability to be completely serious one moment and utterly hilarious the next. Thanks to Ajo Fod for great conversations, sharing the Simpsons, and especially racquetball without any permanent injuries (though I did get really good at being in the wrong place at the wrong time). Thanks also to Richard Vaughan, Paolo Pirjanian, and Andrew Howard for lab spirit and great comments on my talks. Thanks and best wishes to François Michaud for early days at the Interaction Lab.

The are many other people whose encouragement, advice and support have helped over the past five years, including George Bekey, Maria Gini, Jordan Pollack, and Tucker Balch. Thanks also to the wonderful staff whose dedication and help has made the past five years easier and more enjoyable than they otherwise would have been. These folks include: Myrna Fox, Kusum Shori, Julieta de la Paz, Amy Yung, Carol Gordon, Hilda Mauro, Bende Lagua, Laura Lopez, and the rest of the SAL staff. Special thanks to Chuck and Rosario at IS Robotics for helping to keep the R2e's running.

I have saved the last of my acknowledgements for the people whom I can never hope to adequately thank — my family. My deepest love and gratitude to my mother, Monir, the best mother in the world, whose unbound love is the wellspring of my energy. My deepest love and gratitude also to my father, Joel, for always being there when I need him and always on the lookout for things of interest. To my sister, Dori, my warmest love and thanks for dropping by the Lab during my early days at Brandeis, surprising me with food when I was too busy to leave, and in general, always brightening my day. My warmest love also to my brother, Sina, for companionship and shared times too many to mention. Warmest love to Galit and Roii, for friendship and laughs most precious. Many thanks to my beloved relatives in Boston and Los Angeles for all of their love, support, encouragement, and the best home cooking around. In particular I wish to thank: Mamanjoon, Grandma; my aunts, Iran, Simin, Shamsi, Ester, Dina, Dorothy; my uncles, Asher, Bahram, Saied; and all of my other uncles, aunts, and cousins.

The research reported in this dissertation was conducted at the Interaction Laboratory at the Brandeis University Volen Center for Complex Systems and the Computer Science Department, and at the University of Southern California Robotics Research Labs and Computer Science Department. The work was supported in part by the Office of Naval Research Grant N00014-95-1-0759, by the National Science Foundation Infrastructure Grant CDA-9512448, by the Office of Naval Research Grant N00014-00-1-0140, by the Sandia National Labs Grant 3065, and by a grant from the DARPA TASK program.

Contents

List Of Tables

List Of Figures

Abstract

Improving the performance of agent-based systems is a challenging problem requiring both *system evaluation* and *appropriate modification of the agent's policy or controller*. This dissertation presents work in this problem domain, focusing on the development of an on-line, real-time method for modeling the interaction dynamics between a situated agent and its environment. The encompassing theme is to provide pragmatic, general-purpose, and theoretically-sound approaches for improving the performance of agent-based systems.

In order to provide context to the approach and contributions of the dissertation, we first consider some of the many complicating factors that influence a solution to the problem of improving performance. Next, motivation for our on-line modeling approach is provided by a brief examination of off-line evaluation using interference (or collisions) between agents (robots). This work in off-line evaluation presents the unifying experimental theme of the dissertation (mobile robot foraging) and shows how behavior-based control provides a rich substrate for the evaluation of interaction dynamics.

The majority of the dissertation focuses on on-line learning of *augmented Markov models* (AMMs), a novel version of semi-Markov processes. The approach utilizes AMMs to capture agent-environment interaction dynamics in terms of the history of *behaviors* executed while performing a task. These models provide the data that are used on-line and in real-time to evaluate the system and suggest task-dependent, performance-improving modifications to the agent's behavior. An AMM construction algorithm is presented that allows incremental generation with little computational overhead, making it feasible for on-line, real-time applications. The algorithm is able to represent non-first-order Markovian systems in first-order form by dynamically adjusting models through the use of higher-order statistics. This ability to represent higher-order Markovian characteristics provides the expressiveness to accommodate systems with rich interaction dynamics.

The on-line, real-time modeling approach using AMMs in conjunction with behavior-based control is demonstrated as effective in both stationary and non-stationary problem domains. Several challenging robotics applications are examined in the stationary domain (fault detection, affiliation determination, hierarchy restructuring) and the non-stationary domain (regime detection, reward maximization). The AMM-based evaluations used in these applications include statistical hypothesis tests and expectation calculations from Markov chain theory. Experimental results are presented for each of the methods and applications discussed. Finally, some of the statistical distribution issues involving AMMs and their utilization in this work are addressed through an empirical comparison with a non-parametric alternative.

The methods and experimentation presented in this thesis aim to show that the evaluation of agent-environment interaction dynamics can be effective and efficient in improving the performance of agents in challenging problem domains.

Chapter 1

Introduction

This chapter provides an overview of the dissertation and its contributions, placed in the context of key ideas and issues that arise when evaluating the performance of agent-based systems. It establishes the notion of *agent-environment interaction*, and examines evaluation difficulties as a consequence of this interaction, thereby providing perspective on the challenges involved in improving the performance of such systems. This chapter also motivates the remainder of the dissertation by introducing the idea of evaluation using *augmented Markov models* (AMMs) and behavior-based control (BBC).

Agent-based systems are an active area of research in Artificial Intelligence. These systems generally consist of one or more entities, or *agents*, that sense and act within an environment that changes (at least in part) as a consequence those actions. Figure 1.1 illustrates the interaction between

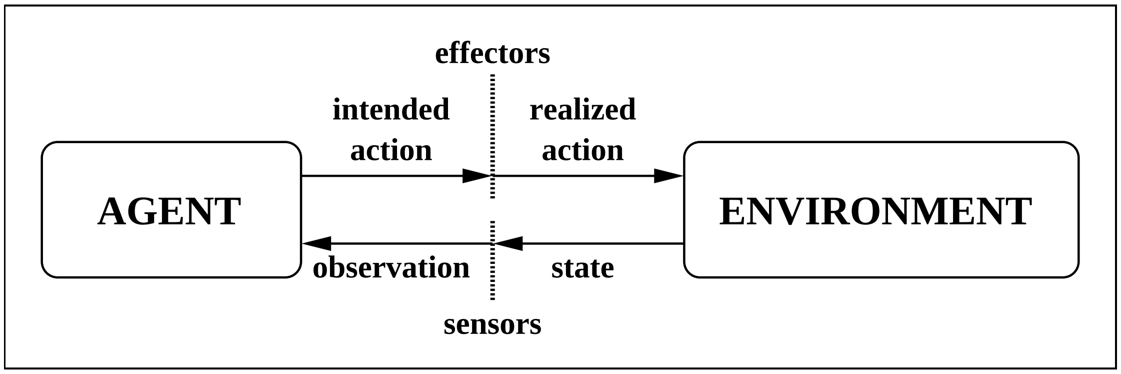


Figure 1.1: The interaction between an agent and its environment.

an agent and its environment. The agent performs actions (mediated through its effectors) that change the state of the environment; the changed environmental state (mediated through sensors) affects subsequent observations and actions by the agent. It is the details of this interaction that determines exactly how the agent performs its function or task. Brooks (1991) argues further that "intelligence is determined by the dynamics of interaction with the world." The concern in this dissertation is not what the richness of interaction implies about intelligence, but rather how it affects performance.

It can be extremely difficult to design a complex agent-based system that initially has optimal (or even efficient) performance. In order to improve performance, two key problems must be addressed, namely:

1. how to evaluate the performance of the system, and
2. how to modify the agent's policy or controller[1] to improve that performance.

Performance optimization is a ubiquitous theme in Artificial Intelligence. In this dissertation, we restate the theme as a general challenge for agent-based systems.

Performance Challenge: **To improve performance through appropriate system evaluation and modification of the agent's policy or controller.**

There are numerous constraints that influence the solution space of this challenge. These include: the sensing capabilities of the agent, the actions it can perform, the function or task it must accomplish, the complexity of the environment, the presence of other agents, and the amount of time available to improve performance. The following sections explore these and other related issues, providing context to the contributions of this dissertation.

In addition, this chapter introduces the major research themes of the dissertation, including the use of machine learning techniques, specifically *augmented Markov models* (AMMs) developed in this work. Throughout the dissertation, AMMs are used in conjunction with *behavior-based control* (BBC), a methodology for constructing agent controllers. The use of AMMs with BBC is further motivated with a brief examination of the author's earlier work exploring the off-line evaluation of multi-robot systems using interference in Chapter 2. That chapter also presents the robot foraging task, the main experimental theme of the dissertation, including the basic behavior-based controller that implements the task and the robots that performed it.

This chapter concludes with a summary of the contributions of the dissertation. The main contribution is the development of an effective method for *on-line, real-time modeling of the interaction dynamics between an agent and its environment, using AMMs and BBC.* The models developed are a foundation for solutions to the *Performance Challenge* above, providing the data

[1]The agent's *policy* provides a mapping from the state of the system (a combination of environmental state as perceived by the agent and the agent's internal state) to actions that the agent performs. A policy ideally tells the agent what action to take in each situation in which it finds itself, so as to accomplish its task as well as possible. The policy is generally encoded as a *controller*, analogous to the way a program codes for an algorithm. A controller often provides a level of abstraction that simplifies and makes concise the specification of the policy.

for the evaluations which suggest application-dependent, performance-improving modifications to the agent's policy. Stated more concisely, the thesis of this dissertation is:

Thesis: **Augmented Markov models, in conjunction with behavior-based control, enable effective evaluation of agent-environment interaction dynamics and facilitate solutions to the *Performance Challenge*.**

The approach is demonstrated in several challenging problem domains involving *embodied* agents (i.e., robots). The link between the contributions of this thesis and the nuances of the *Performance Challenge* will be made clearer in the following sections.

1.1 Issues in Agent-Environment Interaction

When an agent exists in an environment, it is said to be *situated* in that environment, and consequently can interact with it. A subclass of situated agents is *embodied* agents, in which sensing of the environment and actions in the environment are mediated through a "body" (Brooks 1991, Matarić 1999). The body can be virtual, as with an animated video game character, or physical, as with a "nuts-and-bolts" robot. The key notion with both virtual and physical embodiment, however, is that the only sensing capabilities and actions to which the agent has access are those afforded by the body, which therefore provides the physical interface that enables interaction with the environment.

Physically embodied agents are of particular interest in this dissertation, since the experimental domain is physical mobile robots. As we will see below, many of the issues that complicate the *Performance Challenge* for agent-based systems are exacerbated by embodiment.

1.1.1 Sensing and Hidden State

A paradox of sensing is that it can simultaneously provide information that is both excessive and insufficient. The true state of the environment is *hidden* from (or only *partially observable* to) the agent (Whitehead & Ballard 1991, McCallum 1996). Hidden state is an often-encountered issue in situated agent-based systems, though its manifestation is dependent on the sensing capabilities of the agent and the complexity of the task and environment. If these factors are such that the agent can perceive the exact state of the environment, then there is no hidden state. If the possibility of hidden state does exist, the configuration of the environment might make it a non-issue for a particular task. Additionally, sensing is often local, but the agent's movement, including techniques such as active perception (Bajcsy 1988, Ballard 1991), can help compensate for the hidden state associated with the locality of sensing.

Hidden state is more likely to be an issue when the environment is very complex and the sensing capabilities of the agent are relatively impoverished by comparison, as is often true in the domain of physical mobile robots. If the agent's sensing does not allow for full discrimination of the states of the environment, then subsets of environmental state will appear identical, i.e., there will be perceptual aliasing (Chrisman 1992). When hidden state is a factor, discrimination of environmental state must be based in part on a history of sensing. A further complication to state discrimination quite common in mobile robotics is noisy and inaccurate sensing.

These sensing difficulties complicate the *Performance Challenge* by necessitating an evaluation that can suggest policy/controller modifications that accommodate them. Many techniques have been developed that attempt to compensate for hidden state, partial observability, and inaccuracies in sensing. This dissertation assumes that an effective, basic controller for a task can be specified using the appropriate techniques to handle sensing issues. One methodology for constructing controllers is behavior-based control, which is used in this dissertation and discussed briefly in Section 1.3. (Appendix A provides a more in-depth examination of designing and evaluating robust behavior-based controllers for robots.) In regards to sensing difficulties, the focus of this dissertation is on problems that arise during execution and that can be evaluated through their impact on the interaction dynamics between the agent and its environment. As we shall see, monitoring (or "sensing") of interaction dynamics also requires special consideration of hidden state.

1.1.2 Uncertainty of Action

In addition to the sensing difficulties that an agent faces, there are difficulties associated with the actions that it takes (Boutilier, Dean & Hanks 1999). Even assuming that the agent has in its repertoire a set of actions sufficient for accomplishing its task, it does not necessarily mean that it will succeed in doing so. A key problem is that the outcome of actions is uncertain, especially for physically embodied agents. In other words, the action an agent intends to perform can have an outcome that is highly variable.

Consider, for example, a mobile robot that intends to turn 90 degrees in place. If there is no slippage and the robot's mechanical systems are working properly, then it will likely come close to doing so using only open-loop control (i.e., with no sensor feedback). The inherent inaccuracies and noise in the robot's systems, however, make an exact turn nearly impossible. Furthermore, if the floor is dirty, causing the robot to slip, then the turn will be even less accurate. Even though sensing has its own associated difficulties, incorporating it into the turning process (for example with a compass) can help to achieve better control by providing feedback, as we will see in Section 1.3 when we look at behavior-based control. Fortunately, reducing the uncertainty associated with the outcome of actions to an acceptable level can be quite manageable in practice, as will be demonstrated by the foraging task in Chapter 2.

In addition to the uncertainty associated with the outcome of actions, there exists the broader uncertainty of what action is appropriate in a particular situation, relating directly to the *Performance Challenge*. As discussed previously, the agent's policy specifies what action to take in each perceived state of the system. The policy, however, may not be optimal, or even efficient, and thus modification of the policy could improve performance. One approach to policy modification involves the human designer making direct changes to the controller (explored in Appendix A). Alternatively, the agent can *learn* to improve its performance through its experience in executing a task (Section 1.2).

This dissertation is concerned with the uncertainty of action that arises in the natural variability of controller execution for a specific task in a specific environment. We assume that a basic controller can be designed to accommodate much of the uncertainty associated with the outcome of actions. During execution, however, there are likely to be situations (such as hardware failures) that introduce greater uncertainty. In addition, controllers often have decision points where a choice must be made among alternative actions of uncertain appropriateness. The idea in this work is that an agent can learn a model of its interaction with the environment that captures its specific experience during the current execution. This model can then be used to evaluate the system, providing data to reduce uncertainty and choose an action that helps improve performance.

1.1.3 Evaluating Interaction Dynamics

There often exists high variability in the execution of a controller for a particular task by an agent in an environment. In other words, the specific ordering of actions taken by the agent can vary greatly depending on the sensing and action issues discussed above, the exact configuration of the environment, and how the environment is changing. High execution variability increases the difficulty of characterizing the normative behavior of a system. (This helps explain the need for many trials in some of the experiments presented later in the dissertation.) In contrast, execution variability also provides an opportunity to improve performance by taking advantage of the specifics of the current execution.

A notion key to this dissertation is that the execution variability (influenced by sensing and action issues) is captured in the agent-environment interaction dynamics. Modeling these interaction dynamics provides an approach to evaluating the current execution characteristics of the system. This evaluation can then be used in a task-dependent manner to suggest modifications to the agent's policy that are appropriate to the current execution. As an example, consider a robot that has some debris covering one of its sensors. The robot's performance may improve, degrade, or remain unchanged, depending on the sensor affected, the task, and the configuration of the environment. The exact affect on performance is likely only to become apparent during execution as a result of the agent-environment interaction dynamics. Evaluating the dynamics can thus provide a way to assess the system and suggest policy modifications as an approach to the *Performance Challenge*. In this dissertation, the interaction dynamics are captured and evaluated on-line and in real-time using augmented Markov models (AMMs), described in Section 1.2.1.

1.1.4 Stochasticity and Non-Stationarity

Agent-based systems, especially those that are physically embodied, are often replete with noise and uncertainty in sensing and action. As we have discussed, this leads to (potentially high) variability in execution as a result of the agent-environment interaction dynamics. The evolution of these systems is therefore not appropriately described as a deterministic process, but rather as a stochastic (or probabilistic) one. Consequently, probabilistic models, such as the AMMs used in this dissertation, are appropriate for capturing the interaction dynamics of these systems.

An issue orthogonal to stochasticity is (non-)stationarity. In a stationary, stochastic system, the probabilistic characteristics do not change over time. To illustrate this point, let us consider a robot charged with finding a hot cup of coffee. If the configuration of the environment and the robot's controller are stationary (i.e., do not change) then the amount of time it takes the robot to find a cup will follow a particular characteristic probability distribution. The actual time will vary over executions, but the nature of the variability will follow a set pattern. Now, let us consider what happens when the environment is non-stationary, for example, the number of hot cups of coffee decreases over time as they cool or are drunk. There will still be variability in the amount of time it takes to find a cup, but the average time will increase as the number of cups decreases, i.e., the nature of the stochasticity will change.

There are several factors that impact the stochasticity and stationarity of a system. One of these is the structure of the environment. As with the coffee cups, the exact configuration of the environment impacts the stochasticity in the interaction dynamics, and leads to non-stationarity if the configuration is changing. Another factor is learning. As an agent learns, improving its performance by modifying its policy, the nature of its interaction with the environment changes, resulting in non-stationarity. The presence of other agents is also a factor impacting the stochasticity of the system. If these agents are learning or reconfiguring the environment, there will be non-stationarity. In embodied systems, where noise and uncertainty tend to be high, the impact of these factors is exacerbated.

In this dissertation, AMMs are the stochastic model used to capture agent-environment interaction dynamics for solutions to the *Performance Challenge*. We use AMMs in several applications in both stationary and non-stationary problem domains.

1.1.5 Timescale of Performance Improvement

When deciding upon a technique appropriate to meeting the *Performance Challenge* in a particular agent-based system, a key consideration is the time constraint. Perhaps there is no restriction on the amount of time it takes to improve performance, in which case, off-line controller modification can be used (as in Appendix A), or a learning technique that requires experience gained over extended or repetitive execution cycles. Alternatively, the need may be for performance improvement occurring on-line (i.e., during the course of a single execution) and in real-time (i.e., with little computational overhead).

The approach explored in this dissertation focuses on *on-line, real-time performance improvement*. Augmented Markov models are used to learn the agent-environment interaction dynamics during a single execution cycle and provide data used to improve performance during that cycle. Because the time available for improvement is limited, the complexity of the policy modification that can be achieved is also limited. It is not possible, for example, to learn the complete controller for a complex task in a noisy and dynamic environment without extensive time and experience. This work therefore assumes that a basic controller for a task already exists, but that contingencies may arise that require an evaluation of the interaction dynamics, while also providing an opportunity for performance improvement.

In this section, we have considered a number of issues that influence agent-environment interaction dynamics and a solution to the *Performance Challenge*. The next section explores how an agent can improve its performance through learning, and briefly introduces one of the main contributions of this dissertation — augmented Markov models (AMMs).

1.2 Learning in Agent-Based Systems

Learning allows an agent to reach a solution to the *Performance Challenge* without it being provided explicitly by an external expert, such as a human designer. One benefit of learning is the ability to accommodate contingencies in the agent's experience that are not known prior to execution. This relates to the specifics of the agent-environment interaction dynamics arising during execution. One caveat of learning is that careful attention is required to make certain that an appropriate technique is applied to the desired problem. The learning technique should have the correct expressiveness for the task and specific issues surrounding the agent-environment interaction dynamics. In addition, the learning problem should be tractable, and as we will see (in Chapter 3 when we examine some related work in machine learning) there are many techniques that attempt to make complex learning problems more tractable.

In this dissertation, the focus is on learning augmented Markov models (AMMs). The following section provides a brief introduction to AMMs, addressing some of the issues above by showing how AMMs are appropriate to the task of on-line, real-time modeling of agent-environment interaction dynamics. It also places AMMs in the context of other related techniques.

1.2.1 AMMs in Perspective

Augmented Markov models are stochastic models closely related to both Markov chains (MCs) and semi-Markov processes (SMPs), providing a compromise between the two. In Markov chains, the amount of time spent in a particular state follows a geometric distribution, which can be quite limiting since data are often not geometrically distributed. SMPs are a generalization of Markov chains allowing for state durations that follow arbitrary distributions (Ross 1992). In fact, a different distribution can be used for each outgoing transition from a state. AMMs provide some of the generality of SMPs by allowing arbitrary distributions for state durations, but unlike SMPs, AMMs only allow a single distribution for each state. This restriction facilitates the evaluation of AMMs, enabling the use of standard expectation calculations from Markov chain theory. This type of evaluation is crucial in this dissertation, given the aim of using AMMs for modeling and evaluating agent-environment interaction dynamics. In addition, the capacity to represent non-geometric distributions is justified in that the data modeled in the later experimental chapters are generally not geometrically distributed.

Unlike a straightforward SMP representation, the AMM representation of this dissertation incorporates additional statistics in links and nodes which are used during construction and available for evaluation. These statistics allow the AMM construction algorithm, presented later, to dynamically restructure a model to represent, in first-order form, a second-order, or higher-order, Markovian system by maintaining the appropriate order statistics. These statistics are used in conjunction with node-splitting to "unfurl" the higher-order transitions into first-order transitions. Maintaining a first-order representation greatly simplifies many expectation calculations, again allowing standard Markov chain methods to be employed. Node-splitting captures the hidden state associated with the non-first-order nature of a system. In this way, AMMs are related to hidden Markov models (HMMs), though the hidden state captured by HMMs is different in that it is not associated with higher-order transitions (Rabiner 1989). For the purposes of the applications presented in this dissertation, the higher-order representation of AMMs allows capturing interaction dynamics that are non-first order, providing a more accurate evaluation of the system, and consequently, more improvement in performance.

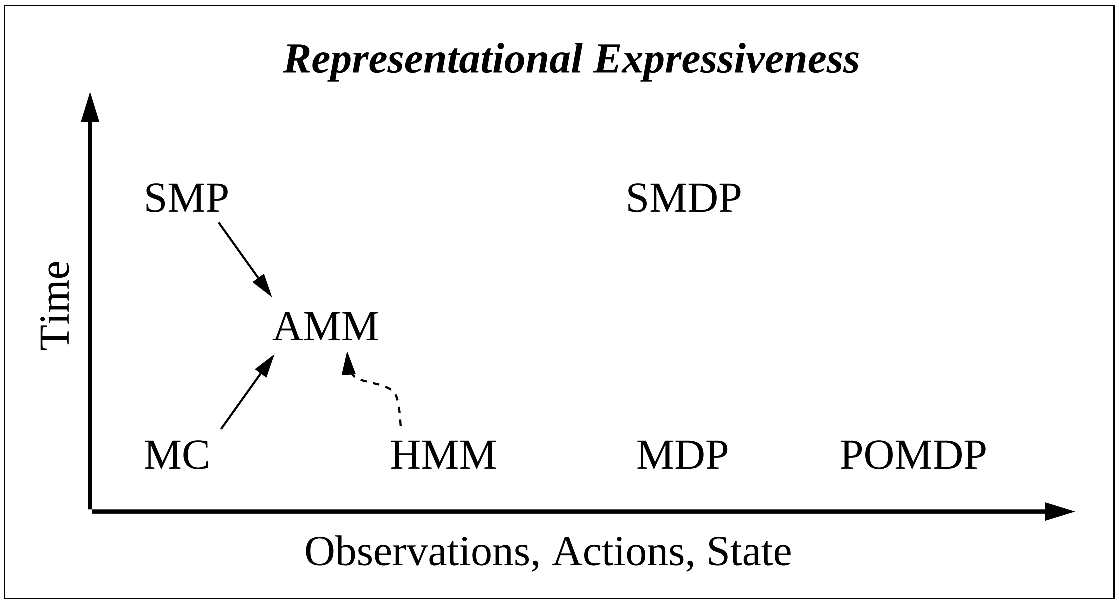


Figure 1.2: The representational expressiveness of AMMs compared to other models. The horizontal axis shows increasing expressiveness in observations, actions, and state. The vertical axis shows increasing expressiveness in time. AMMs share attributes of MCs, SMPs, and HMMs.

A comparison of AMMs and other related stochastic models in terms of representational expressiveness is presented in Figure 1.2. Markov chains are the least expressive of the models, essentially just a stochastic transition matrix (Roberts 1976). As discussed, SMPs are a generalization of Markov chains allowing for a richer representation of time (durations spent in states). Along the horizontal axis, HMMs capture some hidden state, making them more expressive than Markov chains. Unlike HMMs, Markov decision processes (MDPs) allow the explicit representation of actions and their associated rewards, but without hidden state. In addition, each action also has an associated probability distribution over possible outcomes. Partially-observable Markov decision processes (POMDPs) are even more expressive, sharing all of the features of MDPs, but also explicitly incorporating observations and allowing for the possibility of hidden state (Kaelbling, Littman & Moore 1996). Just as SMPs are a more expressive version of Markov chains in time, so are semi-Markov decision processes (SMDPs) to MDPs (Bradtke & Duff 1995, Sutton, Precup & Singh 1999). Figure 1.2 graphically shows that AMMs provide a compromise between SMPs and Markov chains, while also sharing a relationship with HMMs in the ability to capture hidden state.

The question naturally arises as to whether AMMs are appropriate to the problem being explored in this dissertation, namely, on-line, real-time modeling of agent-environment interaction dynamics. We will see in Chapter 4 that the AMM construction algorithm developed in this work allows incremental model construction (enabling on-line application) and, in practice, has low computational overhead and gives real-time response (i.e., with no lag in model construction). The question also arises as to whether AMMs, having no explicit representation of observations (sensing) and actions, are sufficiently expressive to represent the interaction dynamics. The ability to capture higher-order dynamics provides part of the expressiveness required to model the full richness of interaction. The use of behavior-based control (BBC), encompassing both sensing and action, provides the remaining representational expressiveness. Section 1.3 describes BBC and Section 2.4 motivates the use of BBC with AMMs. First, however, we touch on the issue of parametric and nonparametric representations for AMMs.

1.2.2 Parametric versus Nonparametric AMMs

An important factor influencing the appropriateness of a modeling technique to a particular application is the assumptions that the model makes about the structure of the system. One such assumption, mentioned earlier, relates to Markovian order. Standard SMP, MDP, and POMDP implementations assume a first-order Markovian system, whereas AMM construction allows for higher-order systems. A second assumption relates to the probability distributions of the data. Markov chains, HMMs, MDPs, and POMDPs assume geometric distributions for the time spent in each state. By contrast, AMMs, SMPs, and SMDPs allow the possibility of arbitrary distributions. As we will see, there exist tradeoffs between different distribution assumptions.

One dichotomy among distribution assumptions is the parametric/nonparametric distinction. Parametric distributions (e.g., normal/Gaussian, binomial, F, t) allow data sets to be summarized in terms of a few *parameters* that define the exact shape of the distribution. Some parametric distributions, such as the normal, even allow the incremental update of their parameters as new data are added. When the raw data themselves are used to represent the distribution, there is no parameterization involved, and so the distribution is nonparametric.

The advantages of parametric distributions (especially the normal) are that they allow parsimonious representation of the data, and provide the most powerful statistical hypothesis tests when the data conform to the distribution. Unfortunately, the data often do not conform, resulting in conclusions that are potentially very inaccurate. One solution to this problem is to use *robust* versions of parametric statistical tests that can accommodate some degree of non-conformity (Wilcox 1997). A second approach is to use nonparametric statistical tests that make fewer assumptions about the structure of the data (Siegel & Castellan 1988, Hettmansperger & McKean 1998). Disadvantages of nonparametric statistics include the need to retain all of the data, and the fact that they are less powerful than their parametric counterparts when the parametric assumptions are not violated. An advantage of nonparametric tests, however, is that they are usually easy to understand and implement.

This dissertation considers both parametric and nonparametric approaches to representing state duration probability distributions in AMMs. In order to provide parsimony, the majority of the work presented uses parametric AMMs assuming Gaussian state durations. The use of Gaussian distributions also facilitates some of the hypothesis tests used in conjunction with Markov chain expectation calculations in the evaluation of interaction dynamics. In Chapter 8, we will revisit nonparametric AMMs in-depth and compare their effectiveness to that of parametric AMMs.

The next section introduces behavior-based control and its use as the representational substrate for AMMs, providing both expressiveness and a reduced state space for learning.

1.3 Behavior-Based Control

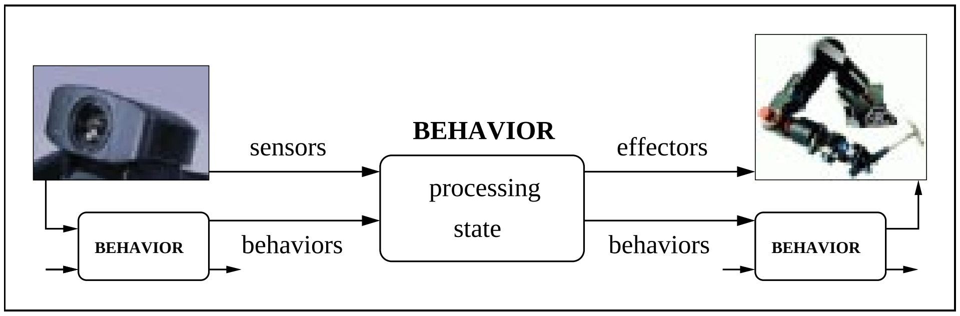


Figure 1.3: The basic structure of a behavior-based controller. Behavior receive input from sensors and other behaviors, process the input possibly changing internal state, and send outputs to effectors and other behaviors. (Sensors are represented by the Sony pan-tilt-zoom camera on the left, and effectors by the Sarcos Dextrous Arm™ on the right.)

Behavior-based control (BBC), a paradigm for constructing controllers for situated agents (Brooks 1991, Matarić 1992), is used extensively in this dissertation. In BBC, a controller is organized as a collection of processing modules, called *behaviors*, that receive input from sensors and/or other behaviors, process the input (possibly modifying internal state), and send output to effectors and/or other behaviors (Figure 1.3). Each behavior generally serves some coherent, independent goal-achieving or goal-maintaining function, such as *avoiding* obstacles or *homing* to a destination. All behaviors in a controller are executed asynchronously and in parallel, simultaneously receiving input and producing output. An action selection mechanism prevents conflicts when signals are simultaneously sent to the same actuators or behaviors (Pirjanian 1998). Behavior-based control has proven to be an effective paradigm for developing single-robot and multi-robot controllers (Matarić 1997*a*, Arkin 1998). Appendix A demonstrates, in detail, the suitability of the behavior-based paradigm for designing robust and modifiable multi-robot controllers.

BBC has several characteristics that make it particularly appropriate to the work in this dissertation, and which justify its use as a substrate for learning with AMMs. First of all, BBC facilitates the accommodation of noisy and inaccurate sensing and action by promoting tight feedback. Noise and inaccuracy tend to be "average out" as the controller frequently senses the world to update actions, in essence adhering to the notion that "the world is its own best model" (Brooks 1991). Because a basic behavior-based controller for a task is able to accommodate much of the low-level noise and inaccuracies, it allows us to model interaction dynamics with a focus on higher-level issues (e.g., the non-stationarity of the environment) that affect the agent's performance.

Behavior-based control also provides the representational expressiveness that is crucial to our use of AMMs for modeling agent-environment interaction dynamics. Since augmented Markov models do not explicitly represent observations (sensing) and actions, the use of AMMs with an appropriate representational substrate is necessary to capture the richness of interaction. BBC, encompassing both sensing and action, provides this substrate. As depicted in Figure 1.4, the combination of AMMs with BBC provides more representational expressiveness of sensing and actions than do AMMs alone.

AMMs with BBC are the synergistic combination integral to this dissertation. AMMs provide the ability for on-line, real-time model construction in higher-order Markovian systems, while BBC provides the representational richness for capturing interaction dynamics. In addition, because BBC abstracts low-level sensing and action into behaviors with coherent functions, it both provides a parsimonious space for AMM construction and facilitates interpretation of the models. This, in turn, facilitates the evaluation of agent-environment interaction dynamics. Throughout this dissertation, we will *use the states of an AMM to represent the execution of individual behaviors of a controller* (Figure 1.5). One caveat in using BBC is that, because the controllers can carry state with extended history that impacts behavior execution, the interaction dynamics captured in terms of behaviors may not be (first-order)-Markovian (Whitehead & Lin 1995). The ability of our AMM construction algorithm to represent higher-order Markovian systems, however, compensates for this.

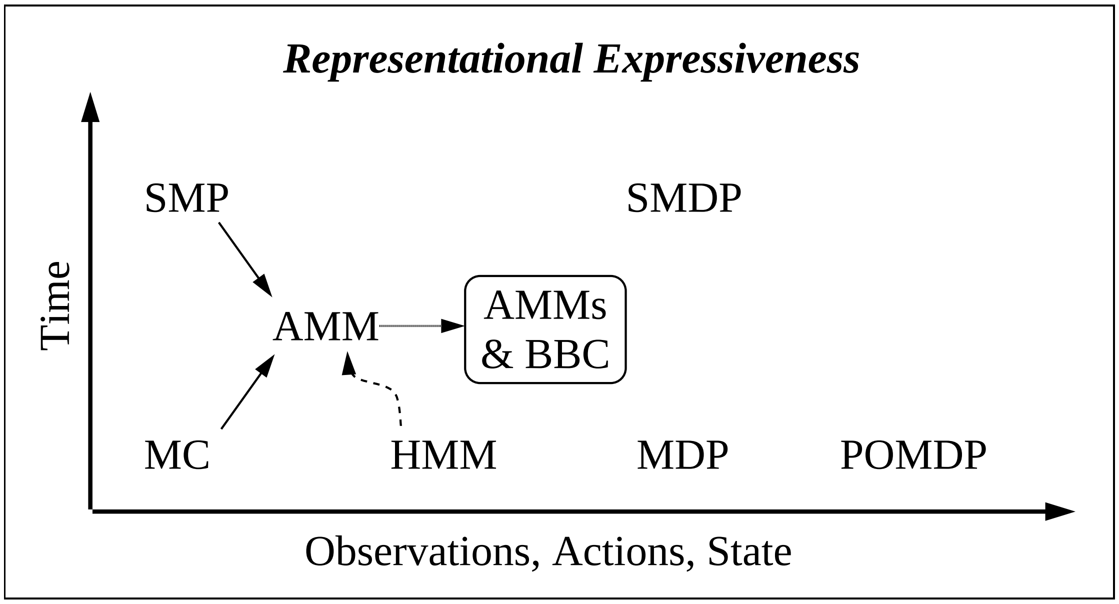


Figure 1.4: The representational expressiveness of AMMs used in conjunction with BBC, compared with other models. The horizontal axis shows increasing expressiveness in observations, actions, and state. The vertical axis shows increasing expressiveness in time. AMMs used with BBC provide a richer representation of observations and actions than AMMs alone.

Chapter 2 presents the experimental theme of the dissertation — mobile robot foraging. It describes the robots that performed the foraging task and the behavior-based controller that implements it. It also presents the motivational result for using AMMs with BBC in Section 2.4. First, however, we review the contributions of the dissertation and outline the remaining chapters.

1.4 Contributions

The contributions of this dissertation are of two types. There are the *main contributions* in direct support of the *Thesis* that augmented Markov models (AMMs) and behavior-based control (BBC) enable effective evaluation of agent-environment interaction dynamics and facilitate performance improvement in both stationary and non-stationary problem domains. There are also the *ancillary contributions* that do not directly support the *Thesis*, but help in the development of the main contributions.

The main contributions of the dissertation are as follows:

- The development of augmented Markov models and their relationship to Markov chains and semi-Markov processes. This contribution includes the representation of AMMs, and the model construction algorithm that uses it to enable on-line, incremental generation with

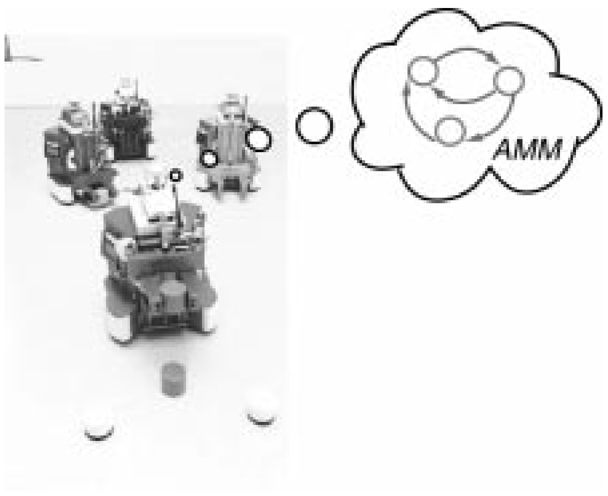


Figure 1.5: Each robot constructs an AMM capturing its interaction dynamics with the environment while performing a task. Each state of the AMM represents the execution of a particular behavior.

dynamic node-splitting for non-first-order Markovian systems. Included in this contribution are also the statistical techniques used to evaluate AMMs.

- The use of AMMs with behavior-based control to capture and evaluate, on-line and in real-time, the interaction dynamics between an agent and its environment.
- The application of this approach to challenging problems involving performance improvement in both stationary and non-stationary mobile robot domains.
- The implementation and experimental evaluation of applications in the stationary problem domain (fault detection, affiliation determination, dynamic leader selection) and the non-stationary domain (regime detection, reward maximization).
- The implementation of a non-parametric version of AMMs and a comparison with the standard parametric version. This contribution includes an extensive empirical study of the statistical distribution issues surrounding the use of AMMs in this dissertation.

The ancillary contributions in support of the main ones are:

- The development of multiple behavior-based controllers for the mobile robot foraging task (the experimental theme), including both individual and group controllers. These controllers are used predominantly with physical mobile robots, but simulated versions are also developed for some of the experimental studies.
- A review of related work. A novelty of this review is a fairly extensive study of computationally efficient approximations from the statistics literature, used in AMM construction and

evaluation. These approximations are widely applicable in Computer Science, and in many cases may be more appropriate than the common techniques. This review aims at increasing awareness of and access to these approximations.

1.5 Dissertation Outline

The remainder of the dissertation is organized into eight chapters and three supporting appendices, as follows.

Chapter 2: Motivation: Mobile Robot Foraging
presents the unifying experimental theme of mobile robot foraging, including a description of the robots that perform the task and the behavior-based controller that implements it. Also presented is the experimental result (from the author's earlier work) that motivated the central approach of this dissertation, namely, modeling interaction dynamics by monitoring behavior execution.

Chapter 3: Related Work
presents a review of related work, focusing on the fields of Robotics, Machine Learning, and Statistics.

Chapter 4: Augmented Markov Models
develops the relationship between Markov chains, semi-Markov processes and augmented Markov models. An overview of the AMM representation and the model construction algorithm is provided, with full details in Appendix B. This chapter also presents the techniques (including Markov chain expectation calculations) used in the evaluation of AMMs, and the details of AMM utilization with behavior-based control.

Chapter 5: AMMs in Stationary Problem Domains
explores mobile robotics applications in the stationary problem domain. Specifically, three applications relevant to group-level coordination are considered: *fault detection*, *affiliation determination*, and *dynamic leader selection*.

Chapter 6: AMMs in Non-Stationary Problem Domains: Regime Detection
examines the use of AMMs in the non-stationary mobile robot problem domain. The focus in this chapter is on detecting significant shifts in the structure of a robot's interaction with the environment that are indicative of environmental regimes, given limited *a priori* knowledge.

Chapter 7: AMMs in Non-Stationary Problem Domains: Reward Maximization
explores a second application in the non-stationary domain. The consideration here is on how a robot can maximize its reward on a task, given it has little *a priori* knowledge of an environment

that is changing.

Chapter 8: Parametric versus Nonparametric AMMs
examines some of the statistical issues associated with the use of AMMs. In the preceding chapters, a parametric version of AMMs is used that assumes normal distributions for state durations. This chapter tests the validity of the assumption through an empirical comparison of parametric AMMs and a nonparametric version that does not assume normal distributions.

Chapter 9: Summary and Future Directions
recapitulates the work in this dissertation and suggests possible extensions.

Appendix A: Design and Evaluation of Robust Behavior-Based Controllers
provides extensive supplementary work on the design and evaluation of foraging controllers, in particular, for groups of robots. This appendix provides a more complete understanding of how to construct and quantitatively analyze a behavior-based controller than do the preceding chapters. This chapter thus complements the earlier ones which assume the existence of a basic controller that can be adjusted on-line to improve performance.

Appendix B: Details of AMM Representation and Construction
gives the complete, unadulterated details of both the parametric and nonparametric AMM representations, and their respective model construction algorithms.

Appendix C: Tables of Critical Points for $\hat{\mathbf{T}}$
provides tables of critical points for the nonparametric test of location described in Section 3.4.1, and used in Chapter 8 and in the construction of nonparametric AMMs in Appendix B.

Chapter 2

Motivation: Mobile Robot Foraging

This chapter motivates the remainder of the dissertation in two ways. First, it presents the unifying experimental theme — mobile robot foraging — including the robots that perform the task, and the basic behavior-based controller that implements it. Second, it briefly examines some of the author's earlier work that inspired the main approach in this dissertation — the evaluation of interaction dynamics by modeling behavior execution.

The behavior-based controller presented in this chapter implements a version of a (multi-)robot foraging (collection) task, a prototype for various applications including distributed solutions to de-mining, toxic waste clean-up, and terrain mapping. We present the general structure of the task, the physical robot test-bed used throughout the dissertation, and the details of the behavior-based controller.

2.1 Foraging Task Structure

We define the foraging task as a two-step repetitive process in which:

1. n robots, where $n \geq 1$, search designated regions of space for certain objects, and
2. these objects, once found, are brought to a goal region using some form of navigation.

A region in the task is any contiguous, bounded space (in the case of mobile robots, a planar surface) which the robots are capable of moving across. There are three mutually-exclusive, non-overlapping types of regions:

- search regions, S, containing a number, p, of objects, a fraction of which must be delivered to a goal region;
- goal regions, G, where objects are delivered;

- and, optionally, empty regions, E, that contain no objects and are not goal regions.

The only restrictions placed on the configuration of regions for the foraging task are: that there be at least one search and one goal region, and that the union of all the regions be contiguous. Figure 2.1 gives two examples of possible valid region configurations for the foraging task.

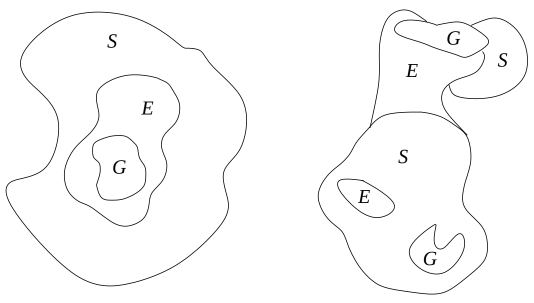


Figure 2.1: Two example region configurations for the foraging task.

The specific configuration used often in this dissertation is shown in Figure 2.2. Experiments are performed in an 11 × 14 foot (or occasionally smaller) rectangular enclosure (the "Corrall"). The search region, S, is approximately 126 square feet and has up to $p = 36$ small cylinders (pucks) evenly distributed throughout. The goal region G, also called *Home*, is a ninety degree sector of a circle with a radius of 2 feet, located in one corner of the Corrall. Finally, there is a 25 square foot empty region, E, separating the search and goal regions. E is composed of the Boundary and Buffer zones, whose functions will be described in the next section. $n \leq 4$ robots are used in the experiments.

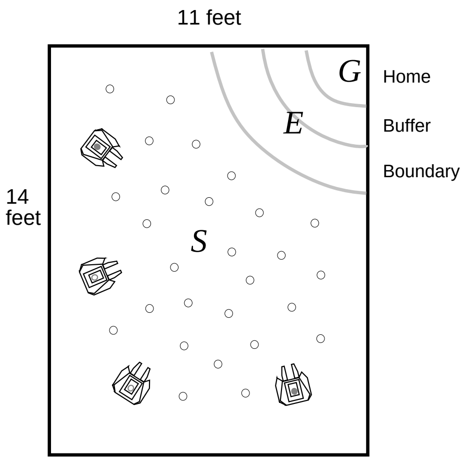


Figure 2.2: One of the foraging task configurations used in the experiments of the dissertation.

2.2 The Robots

Up to four IS Robotics R2e robots are used in the experiments (Figure 2.3). Each is a differentially-steered base equipped with two drive motors and a two-fingered gripper. The sensing capabilities of each robot include piezo-electric contact (bump) sensors around the base and in the gripper, five infrared (IR) sensors around the chassis and one on each finger for proximity detection, a color sensor in the gripper, a radio transmitter/receiver for communication and data gathering, and an ultrasound/radio triangulation system for positioning (Figure 2.4). The robots are programmed in the Behavior Language (Brooks 1990), a parallel, asynchronous, behavior-based programming language inspired by the Subsumption Architecture (Brooks 1986). The main computational power on each robot is a single Motorola 68332 16-bit microcontroller running at 16 MHz. Even though computationally impoverished by today's standards, the processing capabilities have proven to be adequate for most tasks we have envisioned, helping to show that robust, effective control need not be computationally expensive. Perhaps the greatest drawback of the 68332 is its lack of floating point computation, which, for example, influences the calculation of heading, described in the following section.

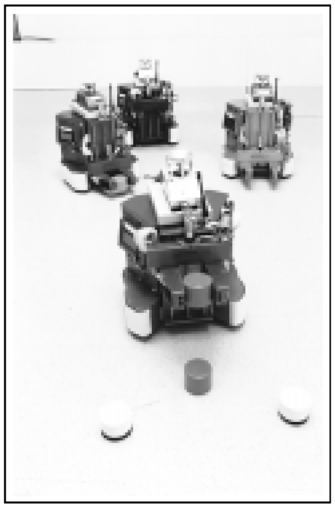


Figure 2.3: The four R2e robots used in the foraging experiments.

The next section presents the basic, *homogeneous* behavior-based controller for the foraging task, used extensively throughout the dissertation.

2.3 The Behavior-Based Foraging Controller

The controller presented in this section performs a *homogeneous* version of the foraging task in which, if there are multiple robots, they all have identical behavioral repetoirs, and act concurrently and independently.

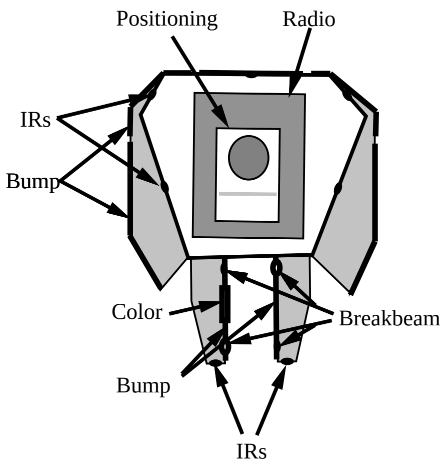


Figure 2.4: The sensor configuration of an R2e robot.

The overall structure of the controller is presented in Figure 2.5. In the figure, the rounded rectangles represent the robot's sensors, with sensor values being transmitted to behaviors along the dotted lines. The behaviors themselves are drawn as ellipses with text in one of three font styles: italics for behaviors that only receive sensor inputs; bold for behaviors that send actuator outputs; and bold-italics for behaviors that do both. The dashed lines represent commands sent by behaviors to the actuators (rectangles), and the solid lines represent control signals sent between behaviors. These control signals include: inhibition signals that temporarily disable behaviors, or do so permanently until the inhibition is lifted; information about the state of the behaviors; and signals indicating that a behavior should perform a certain action. These control signals establish the hierarchy of actuator commands shown at the right of the diagram. The $\otimes$ represents behavior selection and indicates that only one of relevant actuator command pathways is active at any time. The $\odot$ represents a Subsumption-style priority scheme with the actuator command coming from above taking precedence (Brooks 1986). The hierarchy of command pathways in the diagram illustrates that behavior arbitration is the action selection mechanism for the controller (Pirjanian 1998). The next section presents, in detail, the function of the each behavior in the controller, and the structure of the inter-behavior command pathways.

2.3.1 Behavior Details

In order to provide a clear picture of the interaction between behaviors, we describe the individual behaviors of the controller in an order that mirrors the progression of the task as the robot performs it. The following twelve behaviors constitute the foraging task:

1. *avoiding*: This behavior avoids any object (including other robots) detected by the IR sensors and deemed to be in the path of, or about to collide with, the robot. If the robot has already

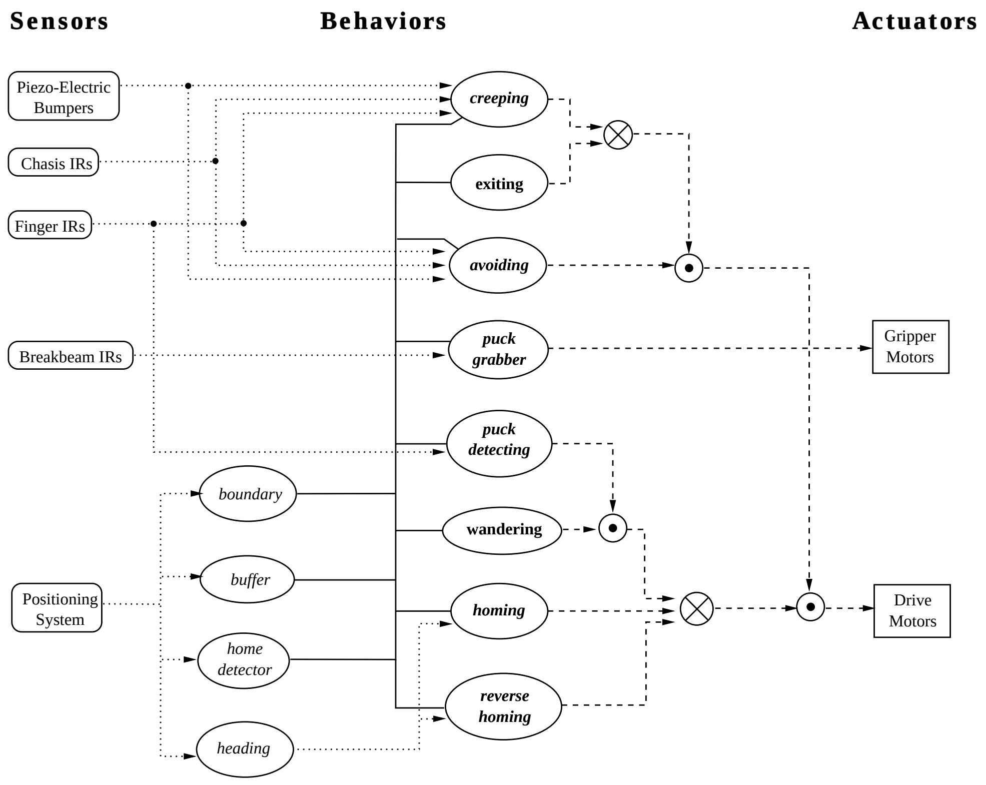


Figure 2.5: The homogeneous controller for the foraging task. Rounded rectangles represent the robot's sensors, ellipses represent behaviors, and rectangles represent actuators. Sensor values are transmitted along dotted lines, actuator commands along dashed lines, and inter-behavior control signals along solid lines. The symbol $\otimes$ represents behavior selection and $\odot$ represents Subsumption-style precedence.

collided with an object, as detected by the contact sensors, it steers away from it. This behavior is critical to the safety of the robot and therefore takes precedence over most of the behaviors that control the drive motors (*puck detecting*, *wandering*, *homing*, *reverse homing*).

2. *wandering*: The robot moves forward and, at random intervals, turns left or right through some random arc. Using this behavior, the robot searches the region for pucks.

3. *puck detecting*: If an object is detected by the front IR sensors while *wandering*, this behavior, by lifting the gripper, determines whether the object is short enough to be a puck, or whether it is an obstacle that must be avoided. If it is a puck, the robot carefully approaches the object and attempts to place it between its fingers. Otherwise, the robot performs *avoiding*.

4. *puck grabber*: When a puck enters the fingers and is detected by the breakbeam IR sensors, this behavior grasps it and raises the fingers. Raising the fingers above puck height prevents the robot from unnecessarily avoiding pucks while *homing*, and allows the robot to collect up to about four additional pucks with its base.

5. *homing*: If carrying a puck, the robot moves towards the designated goal location, Home. While *homing*, *avoiding* can take precedence in order to avoid obstacles.

6. *boundary*: This behavior monitors how the robot enters the Boundary region. If the robot enters this region without a puck, it returns it to the search region using *reverse homing*. If carrying a puck, the robot is allowed to enter this region and proceed towards Home (see Figure 2.2). This behavior prevents the robot from collecting pucks that have already been delivered.

7. *buffer*: This behavior monitors entry into the Buffer region. Entering this region triggers the activation of the *creeping* behavior.

8. *creeping*: A refined combination of the *homing* and *avoiding* behaviors designed to carefully bring the robot to the very corner of the Corrall where Home is located and where the pucks must be delivered. Under *creeping*, the robot moves more slowly and uses its IR sensors at a closer range appropriate for working within the corner. The standard versions of *homing* and *avoiding* would conflict in a confined corner situation, since *avoiding* would perceive the goal corner as an obstacle and attempt to move the robot away from it. *Creeping* takes precedence over *avoiding* since it already incorporates a version of this behavior.

9. *home detector*: A monitoring behavior for entry into the Home region. Upon entering this region, *home detector* sends a signal to *puck grabber* to release the puck.

10. *exiting*: Entering the Home region triggers this behavior which moves the robot several inches backwards, then performs a 180-degree turn in place. This behavior also sends the signal that lowers the gripper. When *exiting* terminates, the robot remains within the Boundary region without a puck. This in turn triggers the *boundary* behavior to begin *reverse homing*.

11. *reverse homing*: Starting from within the Boundary region, this behavior performs the opposite of *homing*, moving the robot out into the search region. This behavior is essentially identical to *homing* except that the goal location is set to the corner of the Corrall opposite Home. Once the Boundary region has been left, *reverse homing* becomes inactive and the robot once again begins searching for pucks using *wandering*.

12. *heading*: This behavior processes the positioning system data and provides approximate heading values for the *homing* and *reverse homing* behaviors. The positioning system supplies the robot's current (x, y) position at approximately 1–2 Hz. Consecutive position values, (x_0, y_0) and (x_1, y_1), are used in an approximate integer-based calculation of $\arctan(\frac{y_1-y_0}{x_1-x_0})$ adjusted for the quadrant of the angle to provide one of sixteen possible sector headings. The accuracy of this heading calculation is usually within one sector of the true heading, but may be far worse when the robot turns in place. Frequent updates of the heading, with little reliance by the other behaviors on any one heading value, help to compensate for the inaccuracies. (An alternative is to use a physical compass for heading data. In our lab, however, the high variance in magnetic fields makes this inviable.)

Now that we have defined the foraging task and the behavior-based controller that implements it, we have the basis for a brief examination of early results that inspired and motivated the notion of modeling interaction dynamics by monitoring the execution of a behavior-based controller.

2.4 Motivational Result: Modeling with BBC

In contrast to this dissertation which focuses on the on-line evaluation of interaction dynamics, the author's earlier work (Goldberg & Matarić 1997) was concerned with the off-line evaluation of the interaction dynamics in a group of mobile robots performing variations of the forging task. In particular, this work showed how interference (or collisions between robots) arising during execution could be used in evaluating multi-robot controllers. This evaluation enabled the use of behavior arbitration schemes (i.e., controller modifications) to adjust the characteristics of interference and the performance of the controllers. As an experimental example of the approach, the work demonstrated three different implementations of the foraging task using the four R2e robots, and presented analyses of data gathered from trials of all three implementations. Many of the details of this work are presented in Appendix A.

One of the key experimental results in this early work, and the initial motivation for this dissertation, was the strong correlation ($\rho = 0.995$) observed between interference and the activation of the *avoiding* behavior (Table A.2). This result seems fairly intuitive, since the *avoiding* behavior is expected to be active during collision events. At the time, however, the strength of the result was a revelation. It begged the question: if a simple measure of behavior activation could capture a key aspect of the interaction dynamics (i.e., interference), would not a more sophisticated model of behavior execution enable a richer evaluation of the interaction dynamics? Further, could not this evaluation be used on-line to enable controller modification and performance improvement during a single execution? The answer to these questions, as is shown in the remainder of the dissertation, is "yes."

2.5 Summary

This chapter has presented the mobile robot foraging task that is the main experimental theme of the dissertation, and to which we will often refer in the following chapters. Also presented was the early motivation for this dissertation and the idea of modeling interaction dynamics using the behavior execution of a behavior-based controller.

Chapter 3

Related Work

This chapter reviews related work, primarily in Robotics, Machine Learning, and Statistics. One of the contributions of this chapter is a compilation of references on computationally efficient approximations to common statistical quantities, and some not-so-common, though excellent, nonparametric tests. These approximations and tests will be used for AMM construction and evaluation in later chapters.

3.1 Robotics, Behavior-Based Control, and Foraging

We begin by considering work related to Robotics and our use of the foraging task. Arkin, Balch & Nitz (1993) demonstrate simulation work studying the issues of density and critical mass in a hoarding task using fully homogeneous robots. Arkin & Hobbs (1993) describe the general schema-based control architecture (which bears some fundamental similarities to the behavior-based control used in this dissertation) and give the critical mass experiments. Finally, Arkin & Ali (1994) present a series of simulation results on related spatial tasks such as foraging, grazing, and herding.

Similar to our behavior-based foraging or mine-collection task, Matarić (1994) describes a behavior-based approach for minimizing complexity in controlling a collection of robots performing various behaviors including following, aggregation, dispersion, homing, flocking, and foraging. The work also includes a simulated dominance hierarchy based on IDs, and used to evaluate performance of homogeneous versus ordered aggregation and dispersion behaviors. There is also a very well explored biological aspect to foraging which has provided some of the inspiration for our experimental foraging task. Gordon (1996) discusses some of the factors that affect task allocation, including foraging, in social insect colonies.

Fontán & Matarić (1998) also worked on multi-robot foraging, but focused on issues of critical mass in task division. Goldsmith, Feddema & Robinett (1998) also present a strategy for multi-robot search, but with each robot able to dynamically switch its team and function. Parker (1992) and Parker (1994) describe multi-robot experiments also on foraging R2e robots with *a priori* hard-wired heterogeneous capabilities using the Alliance architecture. Parker (1994) describes a temporal division that sends one robot to survey and measure the environment for toxic spills, then has the rest of the group use its information to clean up the spill.

Balch (1997) presents Social Entropy, inspired by and based on Information Entropy (Shannon & Weaver 1963), as a metric for describing the heterogeneity of multi-robot systems. As presented, the metric is used off-line to evaluate the learned structure of a robot team. A potential extension of Information Entropy, however, might be to gauge the changing structure of a group on-line.

Cao, Fukunaga & Kahng (1997) offer a view of the multi-agent and multi-robot research applied to 10 ISR R3 mobile platforms, a later generation of our R2e robots. Tan & Lewis (1997) describe an approach to maintaining geometric configurations of a robot group using virtual structures, also tested on the R3's. Similar to our implementation of foraging, this work also exhibits spatial and temporal homogeneity, though the coupling is tighter.

Beckers, Holland & Deneubourg (1994) describe a group of five robots with minimal sensing and no explicit communication effectively clustering pucks through a careful combination of the mechanical design of the robots' puck scoops and the simple controller that moves them forward and in reverse. This work demonstrates a homogeneous controller performing a task similar to our foraging task, but where the goal location is not pre-specified, instead emerging during execution. Holland & Melhuish (2000) present more recent results from an expanded study with essentially the same experimental scenario.

Other work on multi-robot foraging is inspired by trail formation in ants (Drogoul & Ferber 1992). Werger & Matarić (1996) describe a foraging robot chain that is constructed and modified using only contact sensing for communication. Vaughan, Støy, Sukhatme & Matarić (2000) present multi-robot ant-like foraging in a simulated environment where efficient foraging trails are dynamically constructed using a mechanism analogous to ant pheromones.

Behavior-based control has been used in many applications ranging from multi-robot soccer (Lund & Pagliarini 2000) and service robotics (Lindström, Orebäck & Christensen 2000), to control of underwater robots (Rosenblatt, Willams & Durrant-Whyte 2000) and ape-like robots (Hasegawa, Ito & Fukuda 2000). In all of these behavior-based systems, some action selection mechanism produces a coherent, global behavior. The work described in this dissertation uses behavior arbitration in which some (possibly small) subset of the behaviors controls the motors at a given time. Pirjanian (1998) describes a number of action selection mechanisms. Pirjanian, Christensen & Fayman (1998) present a voting-based action selection mechanism which is extended to multi-robot coordination in Pirjanian & Matarić (2000).

3.2 Machine Learning and Robotics

We confine our review to a selection of most relevant work in mobile robotics and statistical modeling. Various models have been employed on mobile robot platforms to date, a few examples of which we consider here. Cassandra, Kaelbling & Littman (1994) use a partially observable Markov decision process (POMDP) to model uncertainty of location in a robot navigation task for an office environment. This work is similar to ours in that both explore model construction on, and use by, a mobile robot. The high computational complexity and large data requirements for POMDP generation, however, make it necessary to use a number of heuristics to reduce the problem size and facilitate learning in addition to often running numerous trials, neither of which is the case with our approach.

In more recent work, Koenig & Simmons (1996) use a POMDP to model sensor, actuator, and metric uncertainties in a similar office navigation task. For tractability, the learning system on the robot is initialized with a POMDP compiled off-line using sensor models and a topological map with metric uncertainties. A modified POMDP learning algorithm is used to passively fine-tune the probability distributions. Somewhat similarly, our work uses AMMs passively to capture statistics about the interaction dynamics between a robot and its environment, with the data used to influence the robot's behavior. Both POMDPs and AMMs represent uncertainties using probability distributions, but POMDPs, being decision processes, also enable learning of control policies. Such policies may require sensor and environment state information, making the search space large and requiring more (possibly heuristic) computation. The number of states in a POMDP is usually pre-specified, whereas in an AMM it is learned based on the order of the system.

Michaud & Matarić (1998) also present a learning technique that uses a behavioral model of a robot's interaction with the environment. The model takes the form of a tree capturing the history of behavior use, i.e., specific sequences of behaviors, with nodes representing executed behaviors, and links representing transitions between them. Their approach and our AMMs are both constructive, being generated and modified as needed. Unlike AMMs, which can form arbitrary graphs with alternative behavior choices represented by probability distributions over transitions, their approach stores potentially long linear sequences of behaviors in a tree with branches indicating alternative choices. The result is that the tree representation may require many trials to collect useful statistics, whereas AMMs can generate them during the course of one trial. This, however, is understandable given the goal of their work is for the robot to learn how to perform a task efficiently. By contrast, in our work the robot has a basic controller for the task, but must make intelligent decisions about how to proceed given its experience in the environment.

Košecká & Bajcsy (1993) present Discrete Event Systems (DESs) with emphasis on applications to mobile robotics. The structure of this DES approach is also related to AMMs, both being represented as directed graphs with behaviors as states. Unlike AMMs, DESs require *a priori* specification of all possible states, events, and the full transition function. This specification endows DES with control theoretic properties, though a practical specification of these parameters for mobile robots would seem to require heuristic engineering. Mahadevan & Theocharous (1998) have applied the notion of discrete (though time-extended) events to a Markov decision process for modeling industrial manufacturing. The goal is to optimize production using reinforcement learning. Other work has also used such hybrid SMP/MDP models, or *semi-Markov decision processes* (Sutton et al. 1999, Wang & Mahadevan 1999), as well as dynamical systems approaches (Beer 1993, Smithers 1995) to model the interaction between an agent (robot) and its environment.

The basic structure of augmented Markov models is very similar to that of hidden Markov models (HMMs) (Rabiner 1989). The difference is that in a AMM, there is only one observation symbol per state, as opposed to a probability distribution over observation symbols in an HMM. In addition, an AMM assumes that the state of the system is known, thereby removing the HMM hidden state assumption. Our construction algorithm, however, is able to capture hidden state associated with higher-order transitions. Han & Veloso (1999) use HMMs to represent robot behaviors in a real-time behavior recognition system employed in a robot soccer domain.

The notion of splitting AMM states based on local estimates of mean and variance is related to work by Hanson on the stochastic delta rule used in updating mean and variance estimates on the weights of feed-forward neural networks. Meiosis Networks use these mean and variance estimates to decide when to split nodes in the hidden layer (Hanson 1990).

Other similar state-splitting approaches have been explored. McCallum (1996) presents Utile Distinction Memory (UDM), an algorithm that splits the states of a POMDP using a method for storing statistics that is almost identical to the statistics in an AMM. The state splitting tests used in the two approaches are also very similar. UDM performs node splitting based on reward distinctions and not perceptual distinctions. This limits the growth of the POMDP to the complexity of the task and not the complexity of the perceptual space. Our use of behaviors (instead of actions, precepts, etc.) serves a similar function — the AMM is only as complicated as the interaction between the robot, controller, and environment, for a particular task.

3.3 Statistics: Parametric Approximations

This section reviews work in Statistics that is relevant to the use of AMMs in the following chapters. In order to maintain low computational overhead in the construction and evaluation of AMMs (Chapter 4 and Appendix B), it is necessary to use computationally efficient approximations to common parametric quantities, including: binomial confidence limits, the cumulative standard normal distribution, the cumulative t distribution, and the cumulative F distribution. Unfortunately, good approximations for these values do not appear to be in wide use. In the hope of increasing awareness, this section reviews some of the literature for these approximations, considering in-depth those that are used in this dissertation.

3.3.1 Binomial Confidence Limits

Let X be a random variable having a binomial distribution, such that

$$Pr(X = x; n, \theta) = \binom{n}{x} p^x (1-p)^{n-x},$$

where n is the number of trials, x is the number of successes, and p is the probability of success (Freund 1992). Given a level of significance, α, we wish to find the upper $(1-\alpha)$ confidence limit, $p^{(1-\alpha)}(x)$, for p such that

$$Pr\{X \leq x \mid p = p^{(1-\alpha)}(x)\} = \alpha, \tag{3.1}$$

for $x = 0, 1, \ldots, n-1$. The confidence interval for p is then $\{0 \leq p \leq p^{(1-\alpha)}(x)\}$, which has an associated infimum coverage probability (i.e., minimum confidence) of $(1-\alpha)$ (Blyth 1986). The symmetry properties of binomial probabilities allow calculation of the complementary lower confidence limit using $p_{(1-\alpha)}(x) = 1 - p^{(1-\alpha)}(n-x)$, for $x = 0, \ldots, n$.

The value of $p^{(1-\alpha)}(x)$ may be calculated from Equation 3.1 using numeric approximation, or exact values may be derived using the inverse beta function, but both methods can be computationally intensive. Tables of pre-calculated values are also available, but, no matter how extensive, they may not cover the desired values of α and n. A more practical alternative is one of the many approximations that use inverted approximations to the normal and F distributions (Blyth 1986). Ghosh (1979) describes two approximations that are quite inaccurate for small values of n (i.e., $n < 30$). Johnson, Kotz & Kemp (1992, pp. 124–133) describe an approximate confidence interval used by the MATLAB® binofit.m function, though this is also computationally intensive. Blyth & Still (1983) presents a table of values for $n \leq 30$ and $\alpha = .95, .99$, to be used in conjunction with a more accurate version of a common approximation when $n > 30$. (Blyth 1986) compares four different approximations, including the Paulson-Camp-Pratt approximation (Paulson 1942, Camp 1951, Pratt 1968), which has the best accuracy for all values of n and x.

The Paulson-Camp-Pratt approximation is among the best approximations available, being often accurate to several decimal places even for very small values of n (Blyth 1986). Thus, it seems quite reasonable for most practical applications, and it is used extensively in the model construction described in this dissertation. The approximation is given by[1]

$$p^{(1-\alpha)}(x) \approxeq \left[1 + \left(\frac{x+1}{n-x}\right)^2 \cdot \left(\frac{81(x+1)(n-x) - 9n - 8 - 3c\sqrt{9(x+1)(n-x)(9n+5-c^2)+n+1}}{81(x+1)^2 - 9(x+1)(2+c^2)+1}\right)^3\right]^{-1} \tag{3.2}$$

where c is the inverse normal cumulative distribution function at $(1-\alpha)$ with $\mu = 0$ and $\sigma = 1$. Table 3.1 provides some precomputed values of c for common values of α. An alternative is to calculate c using one of the approximations cited in the next section. It is important that Equation 3.2 only be used for $x = 1, \ldots, n-2$, when it is difficult to calculate exact values of

[1]The equation as presented by Blyth (1986) has a missing right parenthesis which may cause confusion.

Values of α	Values of c
0.1	1.28155
0.05	1.64485
0.025	1.95996
0.01	2.32635
0.005	2.57583
0.0025	2.80703
0.001	3.09023

Table 3.1: Pre-calculated values, c, of the inverse cumulative normal distribution for use in the Paulson-Camp-Pratt approximation (Equation 3.2).

$p^{(1-\alpha)}(x)$. Otherwise, for $x = 0, n-1, n$, the following easily computed exact values should be used:

$$\begin{aligned} p^{(1-\alpha)}(0) &= 1 - \alpha^{1/n} \\ p^{(1-\alpha)}(n-1) &= (1-\alpha)^{1/n} \\ p^{(1-\alpha)}(n) &= 1. \end{aligned}$$

To calculate $p_{(1-\alpha)}(x)$, x is replaced by $x-1$ and c by $-c$ in Equation 3.2.

Given $p^{(1-\alpha)}(x)$ and $p_{(1-\alpha)}(x)$, the upper $100(1-\alpha)\%$ confidence interval for p is given by $[0, p^{(1-\alpha)}(x)]$, and the lower $100(1-\alpha)\%$ confidence interval is given by $[p_{(1-\alpha)}(x), 1]$. The two-sided $100(1-\alpha)\%$ confidence interval is then $[p_{(1-\alpha/2)}(x), p^{(1-\alpha/2)}(x)]$.

3.3.2 Approximating the Cumulative Standard Normal Distribution

The positive tail probability of a random variable X having a standard normal distribution is given by

$$\Phi(z) = Pr(X \geq z) = \int_z^\infty \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} dx$$

with an inverse $\Phi^{-1}(x) = z$. The positive tail probability is a cumulative distribution, though *the* standard normal cumulative distribution is often considered to be $1 - \Phi(z)$. Given $\Phi(z)$, one may calculate the tail probability at x of a general normal distribution with mean μ and variance σ^2 by setting $z = (x-\mu)/\sigma$. Instead of referring to pre-compiled tables of Φ and Φ^{-1}, which are unlikely to have the accuracy desired and are cumbersome to include in a program, it is often desirable to use one of the many approximations that exist. We briefly discuss a few of the approximations proposed in the literature along with their strengths and weaknesses, then present two of the most accurate.

Page (1977) presents an approximation that is quite accurate for $0 \leq z \leq 2$ with a maximum absolute error of 0.00014, but with a percentage error that grows large for high values of z (Hawkes 1982). Schmeiser's (1979) approximation is not as accurate as that of Page for small values of z, but it is more accurate for larger values, as well as being simpler to calculate. Hamaker (1978) presents an approximation which is not as good as Page's for small values of z, having a maximum absolute error of 0.0061 and a relative error that grows without bound for larger z (Hawkes 1982). A modification to Hamaker's approximation by Lin (1988) is nearly as accurate, but simpler to calculate. A further, though less accurate, simplification is presented in later work (Lin 1990).

Unlike some of this work, the concern here is not in finding a decent approximation that is easy enough to use with a hand calculator. Rather, the desire is to use the most accurate approximation that is not computationally exorbitant. Bailey (1981) makes use of two relatively complicated approximations to $\Phi^{-1}(x)$, one for $0 \leq z \leq 4.98$ and one for $z \geq 4.75$:

$$\Phi^{-1}(x) \approxeq \begin{cases} t_1(1+0.0078365t_1^2-0.00028810t_1^4+0.0000043728t_1^6), & \text{if } x \leq 0.999999, \\ t_2+\frac{0.1633}{t_2^2}+\frac{0.5962}{t_2^3}, & \text{otherwise,} \end{cases} \tag{3.3}$$

where

$$\begin{aligned} t_1 &= \sqrt{-\frac{1}{2}\pi\ln(4x-4x^2)} \\ t_2 &= \sqrt{-2\ln(1-x)-\ln\left(-4\pi\ln(1-x)\right)} \end{aligned}$$

The combination of these approximations is fairly accurate, with a maximum absolute error of 0.00022.

Hawkes (1982) also uses the notion of two complementary approximations, but for approximating $\Phi(z)$. For $0 \leq z \leq 2.2$, Hawkes uses a modification of Hamaker's approximation, and for $z > 2.2$, a minor modification to an approximation in Lew (1981). The combination of these two is remarkable, with a maximum absolute error of 0.000017 and a relative error less than 0.1% for $z \leq 20$. The procedure is as follows:

$$\Phi(z) \approxeq \begin{cases} \frac{1}{2}\left[1-\sqrt{1-e^{(-2t^2/\pi)}}\right], & \text{if } z \leq 2.2, \\ \quad \text{where } t=z-0.0075166z^3+0.00031737z^5-0.0000029657z^7 & \\ \frac{(1.184+z)\frac{1}{\sqrt{2\pi}}e^{(-z^2/2)}}{(1.209+1.176z+z^2)} & \text{if } z > 2.2, \end{cases} \tag{3.4}$$

This approximation is used in the calculation of the cumulative t and F distributions in the following sections.

3.3.3 Approximating the Cumulative t Distribution

The t distribution is useful in determining the statistical significance of the difference between the means of two normally distributed populations, especially when the sample sizes are small (i.e., < 30) (Freund 1992, pp. 406–407). Values for the cumulative t distribution, or its inverse, are generally available in pre-calculated tables or via the widely-used, computationally-intensive incomplete beta function expression of the t distribution. Alternatively, it can be quite desirable to use one of the computationally efficient and quite accurate approximations to the cumulative t distributions.

In general, the cumulative t distribution is approximated via a normalizing transformation, i.e., the t distribution is transformed to allow use of the normal cumulative distribution (see previous section). Prescott (1974) compares normalizing transformations of the t distribution by Anscombe (1950), Quenouille (1953), Chu (1956), Wallace (1959), and Scott & Smith (1970). The approximation by Wallace appears to be the most accurate among these. Mickey (1975) provides a modification of the approximation by Chu. Ling (1978) provides an extremely useful comparison of approximations to the t distribution, including ones by Fisher & Cornish (1960), Gentleman & Jenkins (1968), Peizer & Pratt (1968), and Wallace (1959). The comparison data indicate that an extremely accurate approximation could be constructed using Gentleman-Jenkins up to about 45 degrees of freedom, and Cornish-Fisher at higher degrees of freedom. Considering only a single approximation, the one by Peizer & Pratt provides reasonable accuracy and is easy to program. We now present this approximation in more detail.

Let $Q(t \mid v)$ be the cumulative t distribution with v degrees of freedom. For small degrees of freedom ($v = 1, 2, 3, 4$) the following exact values should be used:

$$\begin{aligned}
Q(t \mid 1) &= \frac{1}{2} - \frac{1}{\pi}\arctan(t) && (3.5)\\
Q(t \mid 2) &= \frac{1}{2} - \frac{t}{2\sqrt{t^2+2}} && (3.6)\\
Q(t \mid 3) &= \frac{1}{2} - \frac{1}{\pi}\left[\arctan\left(\frac{t}{\sqrt{3}}\right) + \frac{t\sqrt{3}}{t^2+3}\right] && (3.7)\\
Q(t \mid 4) &= \frac{1}{2} - \frac{1}{2}\left[1 + \frac{2}{t^2+4}\right]\frac{t}{\sqrt{t^2+4}}. && (3.8)\\
& && (3.9)
\end{aligned}$$

When $v \geq 5$, the approximation by Peizer & Pratt can be used:

$$Q(t \mid v) \approxeq 1 - \Phi\left[\left(v - \frac{2}{3} + \frac{1}{10v}\right)\sqrt{\frac{\log\left(1 + \frac{t^2}{v}\right)}{v - \frac{5}{6}}}\right], \qquad (3.10)$$

where Φ is the cumulative standard normal distribution, such as is given by Equation 3.4.

3.3.4 Approximating the Cumulative F Distribution

The F distribution is used in comparing the variances of two normal populations (Freund 1992, p. 316). Ling (1978) presents a comparison of several normalizing approximations to the cumulative F distribution. The one we focus on here is by Peizer & Pratt (1968).

Let $Q(F \mid v_1, v_2)$ be the cumulative F distribution with v_1 and v_2 degrees of freedom. The Peizer-Pratt approximation (as presented by Ling (1978)[2]) is given by:

$$Q(F \mid v_1, v_2) \approxeq 1 - \Phi \left[d \left(\sqrt{\frac{1 + q \cdot g\left(\frac{S}{np}\right) + p \cdot g\left(\frac{T}{nq}\right)}{\left(n + \frac{1}{6}\right) pq}} \right) \right], \tag{3.11}$$

where

$$\begin{aligned} d &= S + \frac{1}{6} - \left(n + \frac{1}{3}\right) p + 0.04 \left(\frac{q}{v_2} - \frac{p}{v_1} + \frac{q - 0.5}{v_1 + v_2} \right) \\ S &= \frac{v_2 - 1}{2} \\ T &= \frac{v_1 - 1}{2} \\ n &= \frac{v_1 + v_2 - 2}{2} \\ p &= \frac{v_2}{v_1 F + v_2} \\ q &= 1 - p \\ g(x) &= \frac{1 - x^2 + 2x \log x}{(1 - x)^2}, \qquad \text{for x} \neq 1, \text{x} > 0 \\ g(0) &= 1 \\ g(1) &= 0. \end{aligned}$$

Φ is the cumulative standard normal distribution, or an approximation such as is given by Equation 3.4.

3.4 Statistics: Nonparametric Tests

In Chapter 8, we compare the parametric version of AMMs used in much of the dissertation to a nonparametric implementation. One of the key differences between the two versions is the test of location that is used to determine the need for node-splitting. In the parametric version, the location test used is based on the t distribution and assumes normal populations. The idea in the nonparametric version is to use a location test that makes as few distribution assumptions as possible. In the following section, we present a review of nonparametric location tests, focusing on one by Fligner & Rust (1982), which is used in this dissertation.

3.4.1 Nonparametric Tests of Location

One of the most commonly used nonparametric tests of location is the Mann-Whitney-Wilcoxon test (Mann & Whitney 1947, Wilcoxon 1945). It allows greater freedom in the characteristics of the

[2]Due to typographical error, the equation for $g(x)$ as presented in Ling (1978) is missing the parentheses in the denominator, thereby leading to incorrect calculations.

two populations being compared than does the parametric t test. The Mann-Whitney-Wilcoxon test, however, requires that the two populations be symmetric and identical in every respect except location. Thus, the spread, or variances, of the populations can not differ. The Behrens-Fisher problem examines location differences between normally distributed samples from populations that may differ in shape, i.e., have different variances (Fenstad 1983). The nonparametric Behrens-Fisher problem allows non-normal populations, and a further generalization allows non-symmetric populations.

There are nonparametric tests that handle the symmetric version of the Behrens-Fisher problem, including Fligner & Policello (1981). There are also tests for the generalized Behrens-Fisher problem with non-symmetric populations. Hettmansperger & Malin (1975) present one such test, a conservative modification of Mood's (1954) result. Fligner & Rust (1982) present another modification of Mood's test with advantages over the one by Hettmansperger & Malin. Hettmansperger & McKean (1998, pp. 131–133) present a modified version of Mathisen's (1943) test applicable to the generalized problem. In this dissertation, we have chosen the result by Fligner & Rust as an effective nonparametric test of location with very few distribution assumptions. We now provide the details of the test.

Let $X_1, \ldots, X_m$ and $Y_1, \ldots, Y_n$ be independent random samples from populations with continuous cumulative distribution functions $F(x)$ and $G(y)$, respectively. Let θ_x and θ_y be unique medians of the populations, $F(\theta_x) = G(\theta_y) = \frac{1}{2}$. We wish to test the null hypothesis $H_0 : \theta_x = \theta_y$ against the alternative $H_1 : \theta_x > \theta_y$ or $H_1 : \theta_x < \theta_y$ or $H_1 : \theta_x \neq \theta_y$. We define $Z = Z_1, \ldots, Z_N$ to be the sorted list of the X's and Y's, with $N = m + n$, and M as the median of Z. We calculate a version Mood's (1954) statistic, $T = \sum \phi(Y_i, M)/n$, where

$$\phi(a,b) = \begin{cases} 1, & \text{if } a < b \\ \frac{1}{2}, & \text{if } a = b \\ 0, & \text{if } a > b \end{cases}$$

and use this in the modified test statistic $\hat{T} = \sqrt{n}(T - \frac{1}{2})/\hat{\sigma}$. $\hat{T}$ has a limiting normal distribution with a mean of 0 and variance σ^2. An estimate of σ^2 is given by:

$$\hat{\sigma}^2 = \begin{cases} \frac{\frac{1}{4}p(1+p\hat{R}^2)}{(1+p\hat{R}^2)^2} & \text{if } G_n(U_N) - G_n(L_N) > 0 \\ \frac{1}{4}, & \text{if } G_n(U_N) - G_n(L_N) = 0 \end{cases}$$

where $p = m/n$ and

$$\begin{aligned} \hat{R} &= \frac{F_m(U_N) - F_m(L_N)}{G_n(U_N) - G_n(L_N)} \\ L_N &= Z_{(N-b_N)} \\ U_N &= Z_{(b_N)} \\ b_N &= \left\lfloor \frac{N+1}{2} + \sqrt{N} \right\rfloor . \end{aligned}$$

F_m and G_n are the empirical cumulative distribution functions as calculated from the X's and Y's, respectively.

Now, in deciding between the null hypothesis, H_0, and an alternate hypothesis, the following decision criteria are used:

$$\begin{array}{llll}
\text{choose} \quad H_1 : \theta_X > \theta_Y & \text{over} \quad H_0 & \text{if} & \hat{T} \geq \hat{T}_\alpha\{m,n\} \\
\text{choose} \quad H_1 : \theta_X < \theta_Y & \text{over} \quad H_0 & \text{if} & \hat{T} \leq -\hat{T}_\alpha\{m,n\} \\
\text{choose} \quad H_1 : \theta_X \neq \theta_Y & \text{over} \quad H_0 & \text{if} & \hat{T} \geq \hat{T}_{\frac{\alpha}{2}}\{m,n\} \text{ or } \hat{T} \leq -\hat{T}_{\frac{\alpha}{2}}\{m,n\}
\end{array}$$

where $\hat{T}_\alpha\{m,n\}$ is the critical value of the $\hat{T}$ statistic at a significance level of α with sample sizes of m and n. Appendix C provides tables of critical values for $m,n \leq 25$. When $m,n > 25$, the inverse cumulative standard normal distribution at α, $\Phi^{-1}(\alpha)$, provides a good approximation to $\hat{T}_\alpha$ (see Equation 3.3).

One problem with Fligner & Rust's (1982) test is that, even though the test accommodates non-symmetric distributions, the test fails when the distributions are highly non-symmetric, for example when $M = \min(Z)$. The way we have overcome this deficiency in this dissertation is by performing a sanity check, making the alternate hypotheses symmetric

$$\begin{aligned}
H_1 :& \theta_X > \theta_Y \text{ and } \theta_Y < \theta_X \\
H_1 :& \theta_X < \theta_Y \text{ and } \theta_Y > \theta_X \\
H_1 :& \theta_X \neq \theta_Y \text{ and } \theta_Y \neq \theta_X
\end{aligned}$$

and performing all of the associated calculations. When the test fails, at least one term of these alternate hypotheses does not hold, and thus, H_0 is correctly not rejected.

3.5 Summary

This chapter reviewed related work in the areas of Robotics, Machine Learning, and Statistics. In particular, the focus was on single-robot and multi-robot foraging, behavior-based control, and Markov-type models as applied to Robotics. An aim of this chapter (and an ancillary contribution of the dissertation) was to increase awareness of and access to some of the Statistics literature on computationally efficient approximations to common statistical quantities, and less common nonparametric tests. The approximations and test presented in detail will be used in many of the remaining chapters.

Chapter 4

Augmented Markov Models

This chapter presents augmented Markov models (AMMs), and details their relationship to Markov chains and semi-Markov processes (SMPs). It provides an overview of the AMM representation and the model construction algorithm used in this dissertation (with details available in Appendix B). This chapter also lays the foundation for the remainder of the dissertation: it formalizes the use of AMMs with behavior-based control to enable capturing of agent-environment interaction dynamics; and it presents the AMM-based calculations necessary to evaluating those dynamics. The chapter also includes examples of AMM construction.

In Chapters 1 and 2, we introduced and motivated the idea of using AMMs with behavior-based control to capture and evaluate agent-environment interaction dynamics. In this chapter, we provide the key tools and methods for actually doing so in the remainder of the dissertation. We describe the representation and construction of higher-order AMMs, their evaluation, and their use with behavior-based control. First, however, we examine the theoretical underpinnings of AMMs.

4.1 Markov Chains, SMPs, and AMMs

In this section, we develop the relationship between augmented Markov models (AMMs), Markov chains and semi-Markov processes in order to provide theoretical context for our use of AMMs. We begin with Markov chains and work towards AMMs.

A discrete, first-order Markov chain is a stochastic process $\{X_m, m = 1, 2, 3, \dots\}$ with a finite or countable state space adhering to the following property:

$$P\{X_{m+1} = j \mid X_m = i, X_{m-1} = i_{m-1}, \dots, X_2 = i_2, X_1 = i_1\} = P\{X_{m+1} = j \mid X_m = i\}, \quad (4.1)$$

for all states $i_1, i_2, \dots, i_{m-1}, i, j$, and all $m \geq 1$ (Ross 1992). In other words, the probability that the next state X_{m+1} is j, given the current state ($X_m = i$) and any past state ($X_1 = i_1, \dots, X_{m-1} = i_{m-1}$), is dependent only upon the current state i. In general, a stochastic process that satisfies

Equation 4.1 is said to be first-order Markovian. If Equation 4.1 is independent of m, the Markov chain has stationary transition probabilities and is said to be homogeneous. The models described in this section are all homogeneous.

In an nth-order Markov chain, Equation 4.1 takes the following form:

$$\begin{aligned} P\{X_{m+1} = j \mid X_m = i, X_{m-1} = i_{m-1}, \ldots, X_2 = i_2, X_1 = i_1\} \\ = P\{X_{m+1} = j \mid X_m = i, X_{m-1} = i_{m-1}, \ldots, X_{m-n+1} = i_{m-n+1}\}. \quad (4.2) \end{aligned}$$

We assume that Equation 4.2 holds for state j and some $n = n_1 \leq m$, and that for all states other than j the equation holds for some $n \leq n_1$. In other words, the probability of the next state is dependent on the current state and at most $n-1$ previous states.

In a Markov chain, the time spent in each state before a transition to the next state is geometrically distributed. A semi-Markov process (SMP) is a generalization of a Markov chain allowing arbitrary state durations. We let $Q_{ij}(t)$ be the probability of remaining in state i for time $\leq t$ before transitioning to state j. If we let $P_{ij} = Q_{ij}(\infty)$, the P_{ij} define the transition probabilities of the *embedded Markov chain* (Ross 1992) and it follows that $\sum_j P_{ij} = 1$. We let $F_{ij}(t) = Q_{ij}(t)/P_{ij}$ be the conditional probability of remaining in state i for time $\leq t$, given the system has just entered state i and will transition to state j. (Ross (1992) provides further details on Markov chains and semi-Markov processes.)

An AMM is a sub-class of SMP in which the time spent in a particular state is not dependent upon the next state, i.e., $F_{ij}(t) = F_{ik}(t)$ for all states i, j, and k such that $j, k \neq i$. In addition, before a self-transition, the system remains in the current state for exactly one time step, giving:

$$F_{ii}(t) = \begin{cases} 0, & \text{if } t < 0 \\ 1, & \text{if } t \geq 0 \end{cases}$$

Though this formulation makes no assumptions about underlying distributions, the statistical tests we present in this Section 4.3.4 do assume normal distributions. This constrains $F_{ij}(t)$ to be $\Phi(\mu = 1/(1 - P_{ii}), \sigma^2)$, where Φ is the normal cumulative distribution function (with mean and variance determined empirically). Chapter 8 empirically evaluates the violation of this assumption of normality.

AMMs provide a compromise between the generality of SMPs and the computational simplicity of Markov chains. They allow standard expectation calculations from Markov chain theory to be easily combined with popular statistical hypothesis tests, such as the t and F tests, that assume normal distributions. Now that we have placed AMMs in the context of Markov chains and SMPs, we provide an overview of our AMM representation and the construction algorithm that uses it. Full details are available in Appendix B.

4.2 AMM Implementation: Overview

We have seen that an AMM is essentially a semi-Markov process. Unlike perhaps a straightforward SMP representation, our AMM representation incorporates additional statistics in links and nodes which are used during construction and are available for application-motivated evaluations. These statistics allow the AMM construction algorithm to dynamically restructure a model to represent, in first-order form, a second-order, or higher-order, Markovian system by maintaining the appropriate order statistics. These statistics are used in conjunction with node splitting to "unfurl" the higher-order transitions into first-order transitions. Maintaining a first-order representation greatly simplifies many expectation calculations, allowing standard Markov chain results to be employed.

Before continuing, we clarify what an AMM is not. Unlike a Markov decision process (MDP), an AMM does not explicitly represent actions or local reward. There is also no explicit representation of observations, nor the type of hidden-state-inducing partial-observability that is captured in a partially observable Markov decision process (POMDP). In other words, the system is taken to be fully observable with no hidden state of the type in an HMM. The one exception is that the AMM construction algorithm does not assume a first-order Markovian system, but is able to capture, in first-order form, the hidden state arising from the higher-order nature of the system. This differs from the hidden state captured by HMMs and POMDPS which assume first-order systems. The lack of explicit actions and observations in AMMs may seem limiting, but as we discussed in Section 1.3, the use of behavior-based control as a representational substrate provides the expressiveness that compensates for this lack. The simplicity of AMMs also facilitates on-line, incremental, real-time construction and evaluation — a key consideration in this dissertation. In the next section, we present the representational components of an AMM.

4.2.1 Representation of AMMs

For the purposes of conciseness and understandability in this section, we do not describe the full AMM representation as used by the model construction algorithm, but rather, consider only the five-tuple $\langle$ **S**, **A**, **B**, **L**, **T** $\rangle$ containing much of the information necessary for incremental model construction. The details of the elements are as follows:

1. **S**, a set of symbols $\{s_1, s_2, \dots, s_M\}$ recognized by the network. The first symbol, s_1, is recognized only by the first state, a_1.

2. **A**, a set of states (or nodes) $\{a_1, a_2, \dots, a_N\}$. Each state a_i has four attributes:

    - a_i^s, the symbol that the state recognizes, i.e., an element of **S**;
    - a_i^μ, the average number of time steps that the system remains in a_i whenever it enters that state;
    - $a_i^{\sigma^2}$, the variance associated with a_i^μ;
    - and a_i^p, the probability of remaining in a_i in the next time step.

The state, a_1, represents the initial (unknown) state of the system, which is promptly left upon commencement of model construction and never entered subsequently.

3. $\mathbf{B}$, an $N \times M$ transition matrix, where $b_i(k)$ contains the value of the state to transition to if the current state is a_i and symbol s_k is observed. If $a_i^s = s_k$, then $b_i(k) = a_i$, i.e., if the observed symbol is identical to the last symbol observed, then the system remains in the current state.

4. $\mathbf{L}$, a set of directed links $\{l_1, l_2, \ldots, l_P\}$, connecting the states. Each link l_i has the following six attributes:

   - l_i^f, indicates the state from which the link begins, $a_{l_i^f} \in \mathbf{A}$;
   - l_i^t, indicates the state to which the link connects, $a_{l_i^t} \in \mathbf{A}$. The following constraints apply: a link cannot start and end at the same state, $l_i^f \neq l_i^t$; and two links from the same state cannot go to states that accept the same symbol, $\forall i, j$ s.t. $l_i^f = l_j^f$, $a_{l_i^t}^s \neq a_{l_j^t}^s$;
   - l_i^δ, stores the number of times the link l_i has been traversed;
   - l_i^Σ, stores the total number of time steps that the system has been in state l_i^t, after first having traversed the link l_i;
   - $l_i^{\Sigma^2}$, contains the sum of squares of all the durations that comprise l_i^Σ;
   - and l_i^p is the probability of using the link l_i at each time step, given the system is in state l_i^f.

   Because no two links can have the same value for both their *from* and *to* attributes, they cannot represent the same directed transition. Thus, $N - 1 \leq P \leq N(N-1)$: at least $N-1$ links are needed to connect the non-initial states, and for a fully connected network there are $N(N-1)$ links between the non-initial states. The single link from a_1, l_1, is traversed exactly one time, giving $l_1^\delta = 1$ and $l_1^p = 0$.

5. $\mathbf{T}$, a set of structures $\{\mathbf{T}^1, \ldots, \mathbf{T}^{n_{\max}-1}\}$, each with elements $\{t_1^n, t_2^n, \ldots, t_{Q_n}^n\}$ storing information on a particular n-link traversal sequence, where $1 < n \leq n_{\max} - 1$ and $n_{\max}$ is a user-specified maximum order for the model. Each element t_i^n has $n + 4$ attributes:

   - $t_i^{n,1}, t_i^{n,2}, \ldots, t_i^{n,n+1}$, the n links comprising t_i^n, stored as indices into $\mathbf{L}$;
   - $t_i^{n,\delta}$, the number of times the n-link sequence has been traversed;
   - $t_i^{n,\Sigma}$, the total number of time steps that the system has been in the state that link $t_i^{n,1}$ connects to, after first having traversed t_i^n;
   - t_i^{n,Σ^2}, the sum of the squares of all the durations that comprise $t_i^{n,\Sigma}$.

   The bounds of Q_n are given by:

$$\left.\begin{array}{ll} 0, & \text{if } P < n \\ \text{P-n+1}, & \text{if } P \geq 2 \end{array}\right\} \leq Q_n \leq N(N-1)^n.$$

In order for a two-link transition to exist there must be at least two links. If more than two links exist, the fewest n-link transitions ($P - n + 1$ of them) are created when an Euler path exists and is followed through the network. In a fully connected network, each of the $P = N(N-1)$ links has a transition to $N - 1$ other links, giving us the upper bound for a sequence of n links.

Given this AMM representation, the corresponding probabilistic transition matrix of a Markov chain could be generated from a^p, l^p, l^f, and l^t. The addition of a^μ and a^{σ^2} provides the more general (normally distributed) state durations of an SMP. Aside from a^s, the remaining representational elements are used in incremental model generation and dynamic model reconfiguration using node splitting. We provide an overview of the model construction algorithm next.

4.2.2 AMM Construction Algorithm

The data used for constructing an AMM consist of a continuous stream of symbols belonging to $\mathbf{S}$. Construction of an AMM proceeds according to the following algorithm:

- Initialize the system by creating the initial state, a_1.
- If the current input symbol has never been seen before, add it to $\mathbf{S}$ and create a state that recognizes this symbol. Create a link from the current state to the new state and make the transition. Add this transition to $\mathbf{B}$ and create the corresponding new entries for $\mathbf{T}$.
- If the current input symbol is the same as the last input symbol, then remain in the current state. Update the appropriate values in $\mathbf{A}$, $\mathbf{L}$, and $\mathbf{T}$ for mean values and length of time in the state. Recalculate the transition probabilities associated with that state and its links.
- If the current input symbol has been seen before, but is different from the last symbol, transition to a state that accepts the new symbol. If the link for this transition does not exists, create it.
- When transitioning from one state to another, update the variance for the state being transitioned from, and the appropriate sum of squares values in $\mathbf{L}$ and $\mathbf{T}$.
- When about to transition from one state to another using link l_i, do the following:
  1. calculate the binomial confidence interval (Blyth 1986) for the number of traversals ($t^{n,\delta}$) of each t^n which has $t^{n,2} \ldots t^{n,n+1}$ equal to the n links traversed before l_i. If the actual number of traversals that then use l_i as $t^{n,1}$ falls outside the expected confidence interval, then there is an nth-order inconsistency in the traversals. (Section 3.3.1 details the calculation of binomial confidence intervals.)
  2. calculate the t statistic associate with: (1) the mean time spent in the state l_i^f as calculated from the data in each t^n which has $t^{n,2} \ldots t^{n,n+1}$ equal to the n links traversed before l_i, and (2) the mean time spent in the state as calculated from all of the data. If

the t statistic indicates a significant difference, then there is an nth-order inconsistency in the SMP-like state durations.

If either of the previous tests indicates an inconsistency, then the current state is *split*, attaching the current in-link (with its associated out-links, as indicated in **T**) to the new state. This allows an nth-order traversal sequence to be represented in a first-order model. Using **T**, make all appropriate changes to the two states and their related links, in order to keep all global probabilities consistent. Update **T**, modifying and/or adding traversal sequences of the appropriate orders to maintain a consistent model.

The above rules do not provide the complete details, but capture the general flavor of the model construction process. The final rule of the algorithm describes *node splitting* and deserves further explanation. Since an AMM is constructed incrementally as training data become available, it is important that there be some mechanism for model modification when new data invalidate the current structure of the model. This mechanism is provided by node splitting which utilizes data from **T** to attempt to ensure that the model remains consistent with the training data. Node splitting allows an nth-order traversal sequence to be represented in a first-order model, making the model intuitively easier to understand and simplifying expectation calculations with the model.

Note that there is relatively little computation involved in AMM construction when used with behavior-based control. In a non-optimized implementation, the computational complexity per input symbol is at most $O(N^{n_{\max}})$ (for a fully connected network) when a state is being split, and at most $O(N^{n_{\max}-1})$ otherwise. In practice, the use of behavior-based control constrains the complexity to be much more reasonable than what is implied by these values. Because execution of a controller must result in some coherent activity, the possible transitions from any one behavior tend to be small (e.g., $1 \leq N \leq 4$). This essentially brings the maximum computational complexity near $O(1)$, though there might be significant constant overhead. In practice, we have observed that, even for $n_{\max}$ as large as 10, the computational overhead is low enough to allow real-time processing of input symbols from the foraging task at a high frequency (e.g., 100's of Hertz). The space complexity is also $O(N^{n_{\max}})$ for N fully connected states, but again, in practice, graphs tend to be fairly sparse, implying reasonable overhead.

The next section demonstrates the effectiveness of the AMM construction algorithm in a non-first-order Markovian system.

4.2.3 Examples of AMM Construction

Consider the sequence of input symbols $\{3\ 2\ 1\ 4\ 2\ 1\ 3\ 2\ 1\ 4\ 2\ 1\ 3\ 2\ 1\ 4\ 2\ 1 \ldots\}$ that alternates between occurrences of $\{3\ 2\ 1\}$ and $\{4\ 2\ 1\}$. Figure 4.1 gives an example of a first-order or second-order AMM generated with 100 symbols from this sequence. The key item to note in the figure is that from state #4 (accepting symbol 1) there is a 0.5 probability of transitioning to state #2 or state #5, and thus generating $\{3\ 2\ 1\}$ or $\{4\ 2\ 1\}$. We know, however, that this is an inaccurate representation of the system since there can never be two consecutive occurrences of either {3 2 1} or {4 2 1}. Because the next state after #4 depends on two states prior, the system is third order. In contrast, Figure 4.2 shows the first-order representation generated by a third-order AMM constructed with the same sequence. One third-order node split gives two additional states: one that accepts symbol 1 and one that accepts symbol 2. This new first-order representation accurately represents the symbol sequence.

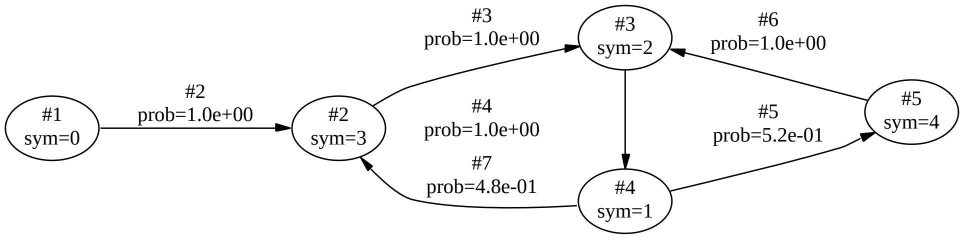


Figure 4.1: A example of a first-order or second-order AMM generated with 100 input symbols from the sequence {3 2 1 4 2 1 3 2 1 4 2 1 . . . }
.

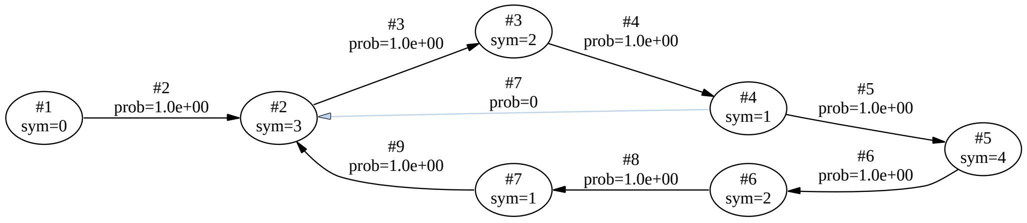


Figure 4.2: A example of a third-order AMM generated with 100 input symbols from the sequence {3 2 1 4 2 1 3 2 1 4 2 1 . . . }
.

This illustration of third-order AMM construction demonstrates the case where there is an inconsistency just in the link traversals, whereas a more subtle case might also involve inconsistencies in the traversal probabilities. A second type of node-splitting occurs based on inconsistencies in the time spent in a particular state. AMM generation for a very complex system may require multiple types of node-splitting at several different Markovian orders.

The next sections details the AMM-based evaluations that will be utilized in the applications of the following three chapters.

4.3 AMM-Based Evaluations

We wish to derive several useful statistics from an AMM which will be used in the following chapters to evaluate the interaction dynamics captured by the models.

One such statistic is the expected number of time steps the system takes to reach a destination state from a given start state. This is known as the *mean first passage*. The theory of Markov chains provides powerful tools for easily calculating such expectations. We apply these tools to AMMs, then use the results to calculate two other statistics: the total variance associated with the mean first passage, and the accompanying degrees of freedom. (A more detailed treatment of Markov chain theory, including proofs and derivations, is available in Roberts (1976) and Kemeny, Snell & Knapp (1966)).

4.3.1 Mean First Passage

The first step in calculating the mean first passage is to extract the $N \times N$ transition matrix P of an AMM M. P is a matrix such that each element p_{ij} is the probability of transitioning directly to state j given the system is in state i. Each element of the matrix is given by:

$$p_{ij} = \begin{cases} a_i^p, & \text{if } i = j \\ l_k^p, & \text{if } \exists k \text{ s.t. } l_k^f = a_i \text{ and } l_k^t = a_j \\ 0, & \text{otherwise} \end{cases}$$

where a_i, a_i^p, l_k^p, l_k^f and l_k^t are extracted from M.

In order to receive meaningful values from the following calculations, P must represent a Markov chain that is *ergodic*, i.e., every state can be reached from every other state. We can determine that a directed graph is ergodic by showing that it consists of one strongly connected component (see Cormen, Leiserson & Rivest (1990, pp. 488–493) for an algorithm).

Given that P is ergodic, we calculate the stationary vector $w = w_1, w_2, \ldots, w_N$ giving the probability of being in each state $a_1, a_2, \ldots, a_N$ of M in the limit. w is found by solving the system of equations $wP = w$ with the constraint that $w_1 + w_2 + \cdots + w_N = 1$. If w has at least one zero value, then P is not ergodic. Using w, we calculate the *fundamental matrix* $Z = [I - (P - W)]^{-1}$, where W is the square matrix having w as each row. W now enters the calculation of the mean first passage matrix $E = (I - Z + JZ_{\text{dg}})D$, where J is a matrix of 1's, Z_{dg} is the diagonal matrix containing the main diagonal of Z, and D is the diagonal matrix with $d_{ii} = 1/w_i$. Each element e_{ij} of E represents the mean first passage from state a_i to a_j.

In our use of AMMs with behavior-based control, we are often interested in the mean first passage from an input symbol s_i to another input symbol s_j. Since each input symbol could be recognized by multiple states in the AMM, the mean first passage between two input symbols might not be unique. In general, if n states recognize s_i and m states recognize s_j, then nm entries in E represent the mean first passage between these symbols. In such a case, our interest is in the *minimum mean first passage between two input symbols* and the corresponding states that give this value in E. For symbols s_i and s_j, let ε_{ij} be this minimum value from E. Let a_α and a_β be the states such that $e_{\alpha\beta} = \varepsilon_{ij}$.

Once we know the states associated with the minimum mean first passage, we can calculate the expected amount of time spent in each state before reaching a_β. To do so, we first convert our ergodic Markov chain into an absorbing chain with a_β as the absorbing state. Essentially, this means modifying P so that once the system enters state a_β it does not leave it, i.e., $a_\beta^p = 1.0$. The transition matrix P is converted into a new matrix Q containing only transitions to and from the $N-1$ non-absorbing states by simply deleting the β-th row and column from P. Note that $\alpha \neq \beta$. The fundamental matrix N for the absorbing chain is given by $N = (I-Q)^{-1}$, where each element n_{ij} represents the expected amount of time spent in the j-th non-absorbing state, given that the system starts in the i-th non-absorbing state, and

$$e_{\alpha\beta} = \begin{cases} \sum_{j=1}^{N-1} n_{\alpha j}, & if \alpha < \beta \\ \sum_{j=1}^{N-1} n_{(\alpha-1)j}, & if \alpha > \beta. \end{cases}$$

4.3.2 Variance of Mean First Passage

In order to calculate the variance associated with a value in the mean first passage matrix, we first note that the underlying Markov chain of an AMM may be interpreted as a set of random variables, one for each state of the chain. Generally, it is assumed that these random variables are independent, identically distributed, and follow a normal distribution. Our use of variance also assumes independence and a normal distribution, but to be more general it allows non-identical distributions.

Given a set of independent random variables $\{X_1, X_2, \dots, X_N\}$ with each X_i having an associated population mean μ_i and variance σ_i^2, we wish to calculate the mean and variance for the linear combination $Y = c_1X_1 + c_2X_2 + \cdots + c_nX_n$. Basic results from statistics tell us that the mean of Y is $\mu_Y = \sum_{i=1}^{N} c_i\mu_i$ and the variance of Y is $\sigma_Y^2 = \sum_{i=1}^{N} c_i^2\sigma_i^2$. In our case, we do not know the population means and variances and instead use the sample means and variances calculated for each state of the AMM, i.e., a_i^μ and $a_i^{\sigma^2}$.

Given $e_{\alpha\beta}$, a_α, a_β, and the fundamental matrix N generated by making a_β an absorbing state, we are ready to calculate the variance associated with $e_{\alpha\beta}$. As shown previously, $e_{\alpha\beta}$ is equivalent to the sum of the row of N associated with a_α. Equivalently, $e_{\alpha\beta}$ may be expressed as a linear combination of the a^μ extracted from the AMM:

$$e_{\alpha\beta} = \sum_{i=0}^{N} c_i a_i^\mu$$

where

$$c_i = \begin{cases} n_{\alpha i}/a^{\mu}_{\alpha}, & \text{if } \alpha < \beta \text{ and } i < \beta \\ n_{\alpha-1,i}/a^{\mu}_{\alpha}, & \text{if } \alpha > \beta \text{ and } i < \beta \\ n_{\alpha,i-1}/a^{\mu}_{\alpha}, & \text{if } \alpha < \beta \text{ and } i > \beta \\ n_{\alpha-1,i-1}/a^{\mu}_{\alpha}, & \text{if } \alpha > \beta \text{ and } i > \beta \\ 0, & \text{if } i = \beta \end{cases}$$

Now that we have expressed $e_{\alpha\beta}$ as a linear combination of the means of the random variables composing the AMM, we can do similarly for the variance of $e_{\alpha\beta}$:

$$\text{var}(e_{\alpha\beta}) = \sum_{i=0}^{N} c_i^2 a_i^{\sigma^2}$$

with the c_i's calculated as above.

4.3.3 Degrees of Freedom

The number of degrees of freedom associated with the linear combination $Y = \sum_{i=1}^{N} c_i X_i$ may be expressed as

$$\frac{\left[\sum_{i=1}^{N} \frac{c_i^2 \text{ var}(X_i)}{V_{X_i}}\right]^2}{\sum_{i=1}^{N} \frac{\left[c_i^2 \text{ var}(X_i)/V_{X_i}\right]^2}{V_{X_i} - 1}},$$

where V_{X_i} is the number of values used to calculate $\text{var}(X_i)$. This is an expanded version of the formula found in Press, Teukolsky, Vetterling & Flannery (1992, p. 617). To apply this to our calculation of the variance of $e_{\alpha\beta}$ we simply use $c_i^2 a_i^{\sigma^2}$ in place of $c_i^2 \text{ var}(X_i)$, with the c_i's calculated as above, and with

$$V_{X_i} = \sum_{\{i | l_i^t = a_i\}} l_i^{\delta}.$$

4.3.4 The t-Test and F-Test

Given the mean first passage values in E and their associated variances and degrees of freedom, we can perform two standard tests of statistical significance: the (one-sample and two-sample) t-test and the F-test (Freund 1992). The t-test uses Student's t distribution to determine whether the difference between the means of two normally-distributed populations is significant. Calculation of this test requires the two mean values and their combined degrees of freedom derived above. The F-test uses the F distribution to determine if the variances of two normal distributions are significantly different given their degrees of freedom. Either of these two tests can be used to compare values for mean first passage and variance within the same AMM and across AMMs. Press et al. (1992) provide more details on the computation of these tests. See Peizer & Pratt

(1968) and Ling (1978) for computationally efficient approximations to F and t tail probabilities for use in these tests.

The next section, describing how AMMs are used with behavior-based control, completes the foundation for modeling and evaluating interaction dynamics in this dissertation.

4.4 AMM Use with Behavior-Based Control

As we have discussed, AMMs can be constructed and utilized on-line and in real-time as an agent (mobile robot) is performing a task, in order to capture the dynamics of its interaction with the environment. At each time step, the datum used for model generation consists simply of a symbol indicating which behavior (or subset of behaviors) of the behavior-based controller is currently active. As we have discussed, the use of behaviors as a representational substrate for model construction has major benefits. It provides parsimony in abstracting away low-level sensor readings and motor commands, while at the same time encompassing the richness of sensing and action manifested in behavior activations.

There is no need to heuristically modify the AMM construction algorithm in order to use behaviors, or provide any application-dependent initialization to the system. What is required, however, is the determination of the ***behavior space***, i.e., the set of mutually-exclusive behaviors, which continuously describe the robot's activity, and are uniquely labeled. One of these labels or symbols (together comprising $\mathbf{S}$) is sent to the model generation algorithm at each time step. Note that the algorithm only accepts one symbol per time step. If the robot's control system is structured so as to utilize simultaneous execution of two or more behaviors, the AMM algorithm will only be able to consider one of their symbols as input. Unless that one symbol is consistently used to represent the parallel execution of both behaviors, the result will be a model unrepresentative of the actual behavior dynamics.

To prevent ill-defined models, it is necessary for the behavior space to be composed of mutually exclusive behaviors which account for all of the robot's activity. In other words, in behavior space $\mathcal{B} = \{\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_K\}$, no two behavior sets $\mathcal{B}_i$ and $\mathcal{B}_j$ may be simultaneously active. In addition, if $\mathcal{T}_{\text{tot}}$ is the total time that the robot is active, and $\mathcal{T}_{\mathcal{B}_i}$ is the total time that behavior set $\mathcal{B}_i$ is active, then $\mathcal{T}_{\text{tot}} = \sum_{i=1}^{K} \mathcal{T}_{\mathcal{B}_i}$. An individual behavior set $\mathcal{B}_i$ in the behavior space may represent several behaviors in the controller that are executed in parallel. The behavior space $\mathcal{B}$ need not contain all of the behaviors that the robot can exhibit, only some composition of behavior sets that meets the preceding constraints.

In the worst case scenario where all of the behaviors of a controller are executed in parallel for $\mathcal{T}_{\text{tot}}$, or when there is only a single behavior in the controller, then $\mathcal{B}$ also consists of exactly one behavior set. The AMM generated in this degenerate case has one non-initial state that it never leaves, and consequently is of no practical use. Fortunately, the large majority of current approaches to robot control, ranging from hybrid to behavior-based systems, utilize sequences and priorities over the actions and behaviors executed on the robot (Matarić 1997*b*, Gat 1998). Thus,

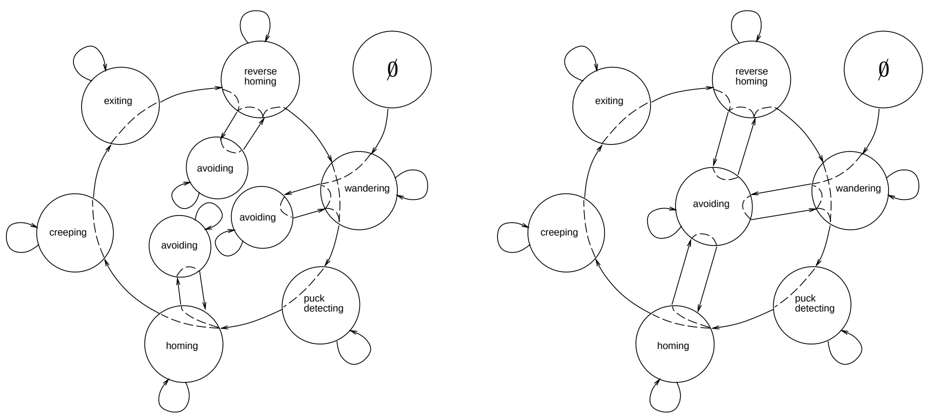


Figure 4.3: (Left) A second-order AMM constructed from foraging behavior data; (Right) A first-order AMM, constructed with the same data.

as long as the behavior of the robot can be decomposed into a sequence of mutually-exclusive behavior symbols to be provided to the AMM, this approach can be applied.

4.4.1 Examples of AMM Construction with BBC

Using the data generated by a robot performing the foraging task (Section 2.3), the AMM generation algorithm constructed the model shown in graphical notation in Figure 4.3 (Left). Many of the numerical details are omitted, but the main elements are present. The states with their recognized symbols (e.g., *avoiding*, *wandering*) are indicated, as well as the initial state represented by $\emptyset$. Links between states are shown, as are the two-link transitions, represented by dashed lines inside states connecting an in-link and an out-link. For the construction of this particular model, we used approximately 1750 data points sampled at 5 Hz over the course of approximately 7 minutes. Only 7 of the foraging controller behaviors were used as the behavior space. All together, the model required 35,000 flops to construct and 2700 bytes to store. The AMM was validated by visual comparison to the robot's behavior while performing the task. Numerous hours of video footage for the foraging task support the validity of the second-order AMM. Consequently, in the following chapters, the AMMs constructed in the experiments that use foraging are all second-order.

The first-order AMM in Figure 4.3 (Right) is not an accurate model of the system. Examining the two-link transitions inside the *avoiding* state, we note that the transition from *wandering* to *avoiding* is followed by a transition from *avoiding* back to *wandering* in all cases. A similar situation holds for *reverse homing* and *homing*. If *avoiding* were really one state, intuitively we would expect transitions between *wandering*, *homing* and *reverse homing* that pass through *avoiding*. These transitions, however, are often inappropriate in the foraging task and seldom occur, again arguing in favor of the second-order model.

4.5 Summary

This chapter presented the tools for capturing and evaluating agent-environment interaction dynamics using AMMs and behavior-based control. Specifically, it outlined the AMM construction algorithm and its associated representation, detailed AMM-based evaluations, and concretized the use of AMMs with behavior-based control. The following three chapters utilize these tools for performance-improving applications in both stationary and non-stationary problem domains.

Chapter 5

AMMs in Stationary Problem Domains

The previous chapters laid the foundation for the modeling and evaluation of agent-environment interaction dynamics using AMMs and behavior-based control. This chapter now utilizes the approach to solve specific application challenges, with a focus on stationary problem domains where the stochasticity of agent-environment interaction dynamics is assumed to be non-changing over time. Specifically, this chapter explores group coordination challenges associated with: individual performance, group affiliation, and group performance. Corresponding respectively to these are the three experimental examples — fault detection, group membership based on ability and experience, and dynamic leader selection.

5.1 Introduction

This chapter examines how modeling interaction dynamics using AMMs and behavior-based control can help improve the performance of a group of agents in the face of contingencies that arise during execution. We consider three issues — individual performance, group affiliation, and group performance — that impact the ability of a group to achieve effective coordination, and present applications of AMMs that help negotiate these issues to improve performance. The assumption in these applications is that the interaction dynamics being modeled are stationary, or any non-stationarity does not significantly impact the application-specific evaluations being used. This is a simplifying assumption that allows us to use a single AMM for evaluation. This assumption is relaxed in Chapters 6 and 7 when we explore non-stationary problem domains.

To help motivate the AMM-based applications of this chapter, we first discuss the three group coordination issues in more detail. As an example of the impact of individual performance on group coordination, consider a scenario where a single robot develops a hardware failure and is neither able to complete its portion of the group task, nor to inform the other group members of its failure. If the members do not know to compensate for the incapacitated robot, the group as a whole may fail to complete its task. Monitoring individual robot performance, in this case for fault detection (one of our experimental examples), is an important component of group coordination.

The ability of a robot to determine what group it belongs to (i.e., its group affiliation) is another important component of group coordination. Suppose a robot were introduced into an environment containing several groups specializing in different tasks. In order to be able to coordinate its activity with the group it fits into best, it must have some mechanism for determining its group affiliation. One way is to compare its abilities and experience with those of other robots — an approach we present in another of the experimental examples.

A third issue impacting coordination is group performance. Consider a group of robots organized in a hierarchy, where the performance of the entire group is strongly dependent on the members in the upper strata. The ability to dynamically reorganize the structure of the group (re-coordinate the individuals) to improve performance is important when unknown or unforeseen circumstances result in poor leaders. We also present experimental results for this type of dynamic leader selection later in the chapter.

In the next sections, we apply AMMs to experimental examples exploring the three issues in group coordination.

5.2 Individual Performance: Fault Detection

For this application, we limit our consideration of faults to those that would keep the robot in one behavior for an inordinate period of time. Such faults may include sensor and actuator failures, as well as the robot becoming physically stuck. To detect a potential fault, we compare, at each time step, the total time the robot has spent in the current AMM state a_i to the mean and variance calculated from previous data for that state. A simple confidence estimate on the upper bound of the mean can be used to make the comparison:

$$\mu = a_i(mean) + c\sqrt{a_i(var)}$$

where c is a small positive constant (e.g., $1 \leq c \leq 3$). If the current time spent in the state exceeds μ, the algorithm signals that there might be a fault.

We tested this algorithm on-line by having the robots perform the *wandering* and *avoiding* behaviors of the foraging task (Chapter 2). Figure 5.1 shows a typical AMM that was constructed. If it was detected that the robot had been in one of the behaviors too long, it would send a signal to the robot, which would in turn beep, thereby indicating a potential fault. We simulated a fault (the robot getting stuck on a rock) by lifting the drive wheels off the ground. During a dozen trials, the robot never failed to detect the fault.

In choosing a particular confidence interval, it is important that one be aware of how rapidly the interval narrows as new data are incorporated into the model. If the interval narrows too quickly, the result will be many false positives, i.e., the robot indicating a fault when none exists. An example of such a confidence interval that narrows too quickly is

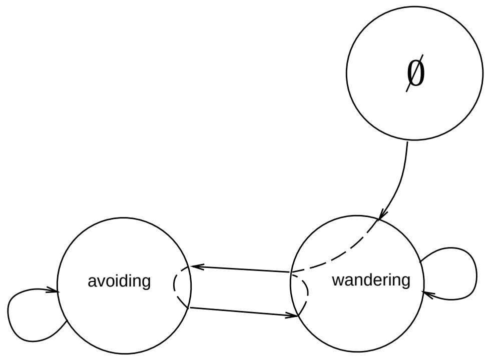


Figure 5.1: Sample AMM constructed from the wandering and avoiding behaviors of the foraging task.

$$\mu = a_i(mean) + t_{.001}\sqrt{\frac{a_i(var)}{n}}$$

where $t_{.001}$ is Student's t distribution leaving 0.001 probability in the tail, and n is the total number of times the system has entered a_i. The first confidence interval may be interpreted as this one, but with $t_p = \sqrt{n}$. This allows us to test for a fault with increasing confidence (i.e., with decreasing values of p) as the number of data points n increases, thus effectively reducing the narrowing of the confidence interval to the rate of variance decrease, and greatly reducing false positives.

Unfortunately, due to limitations in the programming environment of the robots, the models generated for fault detection could not be constructed on the robots themselves. Instead, behavior data were transmitted via radio modem to a Power Macintosh that constructed the models and communicated model information back to the robots. Due to other system limitations, the communication throughput was approximately 2.5 bytes/sec/robot. Even at such rates, and with lost packets, the system was able to maintain useful models simultaneously for multiple robots.

More sophisticated versions of fault detection are also possible using tests based on the mean first passage (Section 4.3). These tests allow the detection of faults that manifest as aberrations in multi-behavior execution sequences and loops.

5.3 Group Affiliation: Membership through Ability and Experience

Coordinating activity in a behaviorally heterogeneous group of robots may require the robots to know their sub-group affiliations. In a learning system where the robot's final behavior is not predetermined, group affiliation is not designated *a priori*. AMMs provide a mechanism for determining group affiliation. Two robots that wish to ascertain whether they belong to the same group can transmit data generated by their AMMs, then determine the probability of the other robot's data on their respective AMMs. The probability of a sequence is the product of the probabilities on the transitions that would be followed to generate that sequence. If a transition does not exist, then the probability is zero. If each AMM accepts the data generated by the other's AMM (with probability >0), then the robots are designated as members of the same group. They are considered to have the same *ability*, or capacity for performing a particular task. In the case of a complex task, such as our foraging example, it may take a significant amount of time for two robot that "should" belong the same group to explore the same interaction dynamics and be determined to actually belong to that group. In the context of our experimental example, there might be several different groups in an area, with only one performing foraging. When a new foraging robot is introduced into the environment, it must determine that its abilities coincide with those of the other foraging robots, and join their group.

In addition to this coarse "don't accept"/"accept", or ability-based, determination of group affiliation, a more refined categorization can be made by considering the actual probabilities of symbol sequences. To test this notion, we ran 2 trials for each of 3 robots performing the wandering-avoiding behaviors. In one trial, the Corrall was empty, in the other, 18 pucks were distributed evenly, though sparsely, on the floor. The robots occasionally avoided the pucks in addition to normally avoiding the walls of the Corrall. For each of the six three-minute trials, an AMM of the robot's behavior was constructed. One thousand data points were generated by each of the six AMMs, and the probability of each data set was calculated on each of the AMMs, resulting in 36 probability values. Our hypothesis was that a data set from an AMM generated in one of the two environments should produce higher probabilities on the remaining two AMMs from that environment than on the three from the other environment. For each data set, we tested all combinations of two AMMs, one taken from each environment, to determine how often the higher probability would be from the same environment. Of the 36 such combinations, in 26 (or 72%) of them, the same environment had the higher probability. These results, produced from little training data and almost identical environments, suggest that AMMs can be used to make subtle behavioral distinctions. These distinctions can be thought of as experience-based. Since the robots are able to and do perform the same task, it is their specific individual experiences that differ, and are the basis for distinction.

More sophisticated variations of affiliation determination are also possible using, for example, calculations of isomorphism between the AMMs of different robots. Another metric might be based on the summed differences between corresponding minimum mean first passage values (Section 4.3). A version of this metric is used in Chapter 8.

5.4 Group Performance: Dynamic Leader Selection

Due to inherent variations in sensors and actuators, or inexperience with a specific robotic platform, it may be difficult to accurately assess the ability of a robot at performing a novel task. Alternatively, even if performance history is available, there is no guarantee that future performance will neither improve nor degrade. Even though the ability of individuals may change over time, it is important that the performance of the group remain as high as possible. To achieve this, some mechanism for dynamic restructuring based on performance is necessary, especially in social structures such as hierarchies where significant reliance is placed on the most dominant individuals. We present dynamic leader selection using AMMs as an example of such a mechanism.

In the following experiments, four R2e robots had to perform the foraging task (Chapter 2). Collecting 10 of the 27 pucks in the Corrall constituted completion of the task, with a shorter completion time corresponding to better group performance. The robots were organized in a strict dominance hierarchy such that whenever two or more robots simultaneously had pucks to deliver to the goal, the most dominant individual was allowed to proceed, while the less dominant individuals each waited their turn. The four robots, however, were not equally efficient at performing the task. The code for each robot was identical, except that the maximum speed was limited to different values, as follows: Robot0 "full-speed" (≈ 0.5 ft/sec); Robot1 "two-thirds-speed" (≈ 0.33 ft/sec); Robot2 "half-speed" (≈ 0.25 ft/sec); and Robot3 "one-third-speed" (≈ 0.17 ft/sec).

We conducted three sets of experiments, two with fixed hierarchies as baselines of comparison to the third, which allowed hierarchy restructuring through the use of AMM-based evaluations. The experiments were designated as follows:

1. **Least Desirable**: The robots were members of a fixed hierarchy with the relative dominance of each inversely proportional to its maximum speed. Thus, Robot3 (the slowest) was the most dominant, and Robot0 (the fastest) was the least dominant.

2. **Most Desirable**: Complementary to Least Desirable scenario, these experiments had the robots arranged in a fixed hierarchy, with the fastest as most dominant, and slowest as least dominant.

3. **Dynamic Leader Selection (DLS)**: The hierarchy was initialized to be identical to that of the Least Desirable experiments, but allowed hierarchy restructuring to improve performance.

In the DLS experiments, with no *a priori* information about a robot's speed provided, an AMM for each robot was constructed at run-time and used to evaluate performance. The metric of evaluation employed was

$$\text{performance} = \frac{\text{number of transitions to } \textit{exiting} \text{ state}}{\text{total time spent } \textit{homing}}$$

where the statistics for the *exiting* and *homing* behaviors came directly from the AMMs. The numerator gives a count of the number of pucks that were delivered to the goal, while the denominator measures the total time spent delivering those. The ratio gives the number of pucks per

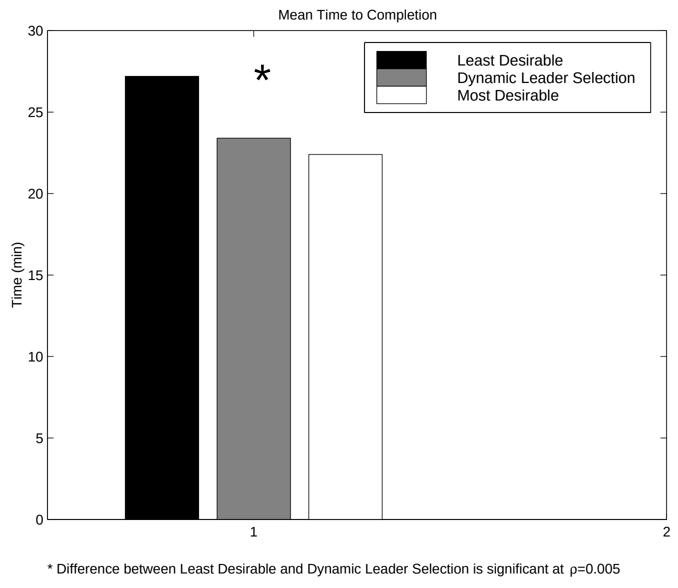


Figure 5.2: Mean time to completion for the Least Desirable, Dynamic Leader Selection, and Most Desirable experimental scenarios.

	Least Desirable	DLS	Most Desirable
Mean time to completion	27.2	23.4	22.4
Standard deviation	1.1	1.3	1.1

Table 5.1: Mean time to completion for the Least Desirable, Dynamic Leader Selection, and Most Desirable experiments.

unit time that a robot is able to deliver: the higher this value, the faster the robot delivers pucks, and the better its performance. (An alternate metric would the minimum mean first passage from *homing* to *exiting*.) Each robot began a trial with its performance value initialized to zero. As it executed the task, its AMM was continuously updated, as was the performance value derived from it. The robot's position in the hierarchy was also updated so that it was more dominant than all other robots with lower performance values.

We ran five trials of each experiment (Least Desirable, Dynamic Leader Selection, Most Desirable) and used a statistical hypothesis test based on Student's t distribution (Freund 1992) to ascertain the significance of our results. In the figures, statistical significance is indicated by asterisks.

Table 5.1 and Figure 5.2 present the average time to completion (i.e., performance) for the three experiments. In the experiments using dynamic leader selection we see a statistically significant improvement in the time to completion over the Least Desirable experiments, thus indicating a successful restructuring of the hierarchies to a more optimal configuration. The Most Desirable

Hierarchy Positions: Dynamic Leader Selection Experiments				
Robot ID	0	1	2	3
Mean position	2.6	1.8	1.4	0.2
Standard deviation	0.9	0.4	1.1	0.4

Table 5.2: Mean positions in the hierarchy at the end of the Dynamic Leader Selection experiments.

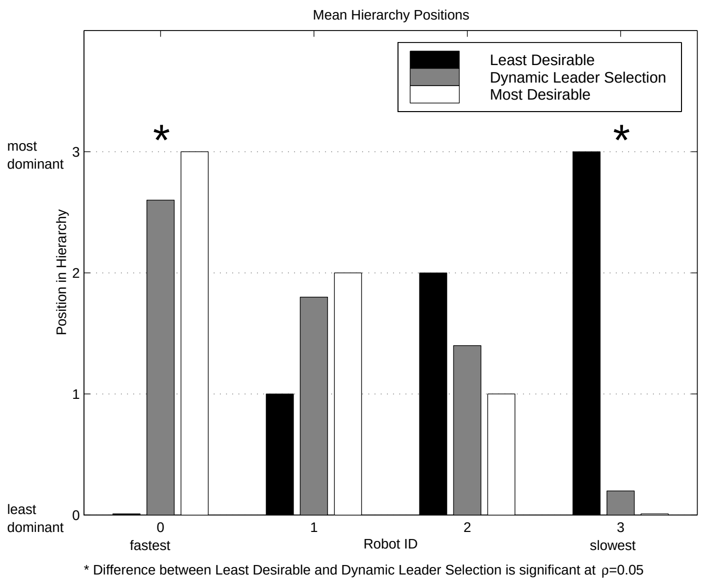


Figure 5.3: Average hierarchy positions at the completion of the Least Desirable, Dynamic Leader Selection, and Most Desirable experiments.

time is slightly, though not significantly, lower than the DLS time. This difference may be attributed to the fact that the DLS experiments are initially configured with the less efficient Least Desirable hierarchy structure.

The successful restructuring of the hierarchies is evident in Figure 5.3 and Table 5.2. We see that the final hierarchy positions in the Least Desirable and Most Desirable experiments are unchanged from the initial positions since no hierarchy restructuring was allowed to take place. Even though the initial hierarchy positions in the DLS experiments are identical to those of the Least Desirable experiments, we see in Figure 5.3 that by the end of the trials, the positions are almost identical to the Most Desirable ones, though always lying between the Least Desirable and Most Desirable.

The fact that the average DLS positions are not exactly equal to the Most Desirable positions (indicating non-optimal restructuring) requires further explanation. At the beginning of each DLS trial, when all of the robots have performance values of zero and the hierarchies are still identical to the Least Desirable, it is very likely that one or more of the slower, more dominant robots will

Pucks Collected												
	Least Desirable				*DLS*				*Most Desirable*			
Robot ID	0	1	2	3	0	1	2	3	0	1	2	3
Mean number of pucks collected	1.2	2.6	3.2	3.0	3.2	3.4	2.0	1.4	3.8	3.2	2.2	0.8
Standard deviation	0.4	0.5	0.8	1.0	1.3	0.9	1.2	0.5	0.4	0.8	1.1	0.4

Table 5.3: The mean number of pucks collected in the Least Desirable, DLS, and Most Desirable experimental scenarios.

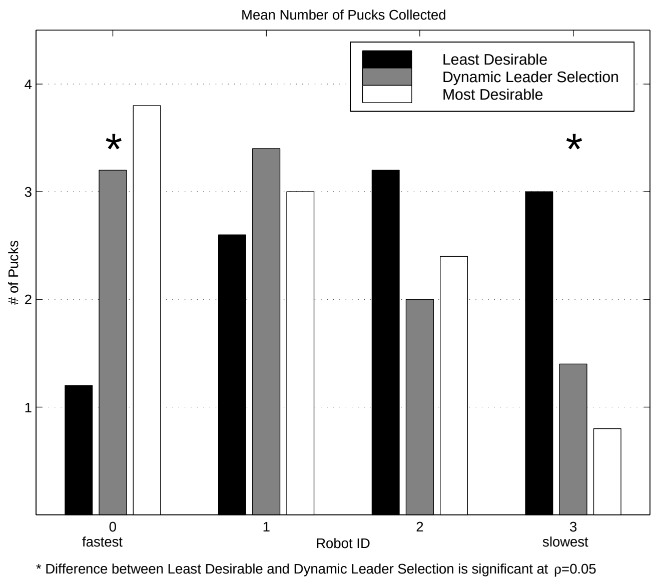


Figure 5.4: Mean number of pucks collected at the completion of the Least Desirable, Dynamic Leader Selection, and Most Desirable experiments.

find a puck and be allowed to deliver first, thereby attaining a non-zero performance value. This helps establish its position in the hierarchy and makes it more difficult for the faster, less dominant robots to get a chance to deliver a puck and get a better performance value. In addition, the foraging task is quite stochastic. On average, all of the robots will find pucks the same distance from the goal, but in any one trial, a slower, less capable robot may find pucks very close to the goal and deliver them in little time, thereby gaining a higher performance value than a more capable robot finding pucks further away. Similar to the fault detection experiments, the AMMs in these experiments were also generated on a PowerMac with data transmitted to and from the robots via radio modem. Lost data packets and noisy signals, in addition to occasional robot failures, also hindered optimal restructuring. With all of these complications, the success of the Dynamic Leader Selection experiments is an indication of the robustness of the approach presented here.

Figure 5.4 and Table 5.3 present the average number of pucks that each robot collected in each of the three experiments. We note by a comparison with Figure 5.3 that in all three experiments, the number of pucks a robot collected is proportional to its final position in the hierarchy. Intuitively, this is consistent with the notion of a dominance hierarchy: the less dominant a robot is, the less often it will be allowed to deliver a puck, since it must wait to be the most dominant individual ready to do so. In a large hierarchy, this may never happen.

There are two slight anomalies in the puck data that, although not significant, bear some consideration. In the Least Desirable experiments, Robot3 (the most dominant and slowest) did not collect as many pucks as Robot2, although one might expect it to collect more. One explanation for this inconsistency is that the performance of Robot3 degraded more than could be compensated for by its high position in the hierarchy. Robot3 was simply poor at finding pucks. There is a complementary inconsistency in the DLS data where Robot0 (the fastest, and on average most dominant at the end of the trials) collected fewer pucks than Robot1. This seems due to the fact that Robot0 began the DLS trials as the least dominant robot, and consequently at times was not allowed to deliver a puck and establish its performance value until relatively late in a trial.

Finally, it is worth noting that none of the results for the Most Desirable experiments were significantly different from the dynamic leader selection experiments. One implication of this result (and some of the others mentioned above) is that in order to test the significance of minor differences in the data gathered, additional, possibly numerous, trials would be necessary — likely an impossibility due to time and robot robustness constraints. The complementary implication is that, on average, dynamic leader selection using AMMs yields performance virtually identical to a pre-specified optimal hierarchy, even starting with a very undesirable hierarchy.

5.5 Summary

This chapter presented three experimental applications focusing on group coordination issues and assuming stationary interaction dynamics. The evaluation of agent-environment interaction dynamics using AMMs and behavior-based control was shown to be effective in fault detection, affiliation determination, and dynamic leader selection — all important to group-level performance. The next chapter explores applications in non-stationary domains.

Chapter 6

AMMs in Non-Stationary Problem Domains: Regime Detection

This chapter relaxes the assumption of stationarity in Chapter 5, and focuses specifically on detecting significant changes in the agent-environment interaction dynamics. It presents an approach using multiple AMMs to monitor events at different time scales and provide statistics to detect changes at those time scales. The approach is successfully implemented using a physical mobile robot performing a land mine collection task (a variation of foraging), and experimental results are provided.

6.1 Introduction

In certain classes of tasks, it may be necessary for a situated agent to detect significant global changes in the environment and modify its behavior or the task structure accordingly. The environment can be in a particular regime (i.e., a period of steady state) and then switch to a different regime requiring the agent to modify its behavior. Detecting such environmental regime changes may be difficult for a number of reasons:

- The agent may have no *a priori* knowledge of the environment and thus also lack a baseline for gauging environmental shifts. In a system where the environment is evolving (i.e., a non-stationary system), determining a basis for comparison may be difficult.

- Given only local sensing capabilities, the agent may require a significant amount of time to estimate the state of the environment. Any estimate of state, however, may be outdated in a non-stationary system.

- The nature of the task may be stochastic, with uncertainties large enough to preclude an effective predictive model of environmental state, or dynamics too complex to make the development of such a model feasible or tractable. Alternatively, however potentially simple the system, there may be no *a priori* data with which to instantiate a model.

- Depending on the task or environment, the time scale of the environmental non-stationarity that must be detected may differ. For example, in one task, the environmental change may be almost instantaneous, detectable between one moment and the next. In another task, the change may be slow and incremental, requiring the examination of a large time interval for detection. Hard-coding the agent with a specific time scale to use for regime detection can be problematic. A time scale that is too small makes the robot incapable of detecting the change. Conversely, a time scale that is unnecessarily large increases the time required to detect the change and may be undesirable in time-critical situations.

As a concrete example, consider the task of collecting undetonated land mines in a field. Assume that there are two types of mines, large and small, with destructive power proportional to their size. A robot is given the following instructions: "Go out to the field and first collect as many large mines as you can, since they are the more destructive. But don't spend all of your time searching for every last large mine if you discover that there aren't many of them. Instead, start collecting the small mines. After all, we want to clear the field as well as possible." In order for the robot to accomplish this task, it must have enough data about its environment (the mine field) to intelligently switch from collecting large mines to small ones. In this scenario, the robot is only able to carry one mine at a time, producing a large cost (in time) for each mine collected. It is important that the more critical large mines be collected first, but that the robot be able to decide when to switch to the smaller mines. (Here we assume that the task requires the robot to collect one type of mine at a time. Alternatively, the robot might switch between types as necessary. We explore this alternative when we consider a reward maximization scenario in Chapter 7.)

The difficulty of this task is compounded when the issues mentioned above apply. The robot may have no *a priori* information about the numbers of large and small mines in the field, their distributions, or relative proportions. The robot may also lack global sensing of the mines in the field and may not know the time scale appropriate to its decision for switching between mine types. This decision is dependent on factors including the size of the field and the relative densities of the two types of mines.

In this chapter, we propose a mechanism for regime detection that resolves the above issues. The approach uses multiple AMMs to capture, in real time, the dynamics of a robot interacting with its environment in terms of the behaviors it performs. One AMM is created and maintained at each time scale that is monitored, and statistics about the environment at that time scale are derived from it. As task execution continues, AMMs are dynamically generated to accommodate the increasing time intervals. Sets of statistics from the models are used to determine whether the environmental regime has changed. This approach requires no *a priori* knowledge, uses only local sensing, and captures the notion of time scale. Additionally, it works naturally with stochastic task domains where variations between trials may change the most appropriate time scale for regime detection. The approach has been physically realized on a mobile robot performing the mine collection task. Experiments and results for this task are presented later in the chapter.

It should be noted that it is difficult to define an absolute notion of regimes, especially since it relates to the dynamics of environmental changes. In a gradually shifting environment, the designation of a regime change can be fairly arbitrary. In this chapter, we propose one principled method for regime detection based on statistical hypothesis tests, and empirically show it to be effective.

In the next section we describe how AMMs may be used as part of a mechanism for regime detection.

6.2 AMMs for Regime Detection

Our focus is on the difficult but realistic situation in which a robot lacks *a priori* information about its environment, an environmental model, and global sensing. In such a situation, the robot may require a relatively large amount of time to detect a trend that signals a global environmental regime change. This is especially so if the system is noisy and stochastic, as is generally true for mobile robotics. Unless a sufficiently large time scale is employed, the regime change may be lost in the variation of the data. Determining the appropriate time scale, however, may not be possible ahead of time. It may be dependent on the exact nature of the task, the structure of the environment (including the presence of other robots), and the nature of the system's stochasticity. The time scale may also dependent on the specific attribute(s) of the system being monitored for regime changes.

In order to negotiate these challenges and endow a robot with the ability to detect global environmental regime changes, we maintain models (AMMs) of the robot's interaction with its environment at multiple time scales. As the robot performs its task, we extract and store particular statistics from these models, which are used to detect a specific regime change based on a sound criterion of significance. In our first experimental validation, the regime switch is detected as a significant change in the density of mines, while in the proportion-maintaining scenario it is detected as a change in the proportion of mines. Before presenting the algorithm for regime detection, we first introduce some notation used in the algorithm.

6.2.1 Notation

- Let $\tau > 0$ be the minimum time scale, or number of input symbols, used to construct an AMM.
- Let f_i be a positive valued function of τ returning the size of (number of input symbols maintained by) the i-th time scale.
- Let $k > 0$ specify the number of AMM-extracted values used in detecting a regime.
- Let the AMM at time scale f_i be m_i.
- Let Q_i be a sequence of at most k statistics for model m_i.
- Let n be the total number of input symbols that have been used to construct the models.
- Let M be a special AMM that is constructed using all of the input symbols that have been seen.

This notion is now employed in the regime detection algorithm.

6.2.2 Algorithm for Regime Detection

1. Initialize M, m_0, and set $n \leftarrow 0$.
2. Get an input symbol and use it to update M and all m_i.
3. Set $n \leftarrow n + 1$.
4. For all i such that $(n \bmod f_i) = 0$
   (a) If no such m_i exists due to the fact that a new time scale has been reached, then create m_i and initialize it to equal M.
   (b) Call Stat(m_i) to get the statistic for the model and insert that value into Q_i.
   (c) If the length of Q_i equals k, then call DetectRegime(Q_i).
   (d) If DetectRegime(Q_i) returns true, then the regime has changed, else it has not.
   (e) Re-initialize m_i to be an empty model.

Stat() is a function on an AMM, returning application-dependent statistics extracted from the model (e.g., the mean time in a state/behavior). DetectRegime() performs a statistical hypothesis test (such as Student's t or ANOVA) on a list of values. DetectRegime() returns true if the result is significant, false otherwise. Essentially, DetectRegime() provides a *meta-threshold* based on a statistical hypothesis test. The threshold is not a set number, but rather a measure of the statistical significance of the shift in the environment.

The algorithm maintains multiple AMMs at different time scales. At each time step, each AMM (m_i and M) is updated with a new input symbol. If no m_i exists for a new, larger time scale (f_i), then that model is created and initialized to M. If a model m_i has received its maximum number of input symbols (as designated by f_i), then Stat(m_i) is called to extract the appropriate application-dependent data from it, and m_i is reinitialized to be empty. The data from m_i is inserted into a queue Q_i of maximum length k, and if $\mid Q_i \mid = k$ then DetectRegime(Q_i) is called to test for significant differences in the values of Q_i. It is just such a significant difference or shift in the data which is designated as a regime change.

In the next section we describe our experimental setup and example.

6.3 The Land Mine Collection Task

To validate our algorithm for detecting global environmental regime changes, we use a task analogous to the land mine collection example from the beginning of the chapter, which is also a version of foraging from Chapter 2. We use one R2e robot (Figure 2.3) in the experiments. The Corrall is adjusted to be either 11 × 14 feet or 11 × 8 feet, depending on the experiment, and contains up to 36 pucks of two different colors: clear (representing large mines) and black (representing small ones). Figure 6.1 shows two experimental configurations. The behavior space (Section 4.4) for this task consists of the following nine behaviors:

- *avoiding*: avoid any object (detected by IR and contact sensors) deemed to be in the path of the robot.
- *wandering*: move forward and, at random intervals, turn left or right through some random arc.
- *puck detecting*: if avoiding is not active, and if an object is detected by the front IRs, lift up the gripper fingers to determine whether the object is short enough to be a puck. If it is, approach the object and try to place it between the fingers and pick it up. If unsuccessful, perform *avoiding*.
- *color detecting*: if *puck detecting* is successful, detect the color of the puck. If it is the desired color, then perform *homing*, else perform *leave puck*.
- *leave puck*: drop the puck and continue searching for more, using *avoiding*, *wandering* and *puck detecting*.
- *homing*: if carrying a puck, move towards the designated goal, Home.
- *creeping*: when near Home, perform a slower, more accurate homing behavior.
- *exiting*: if in the Home region, drop puck and exit Home.
- *reverse homing*: move away from the Home region.

The two behaviors that are new to the land mine collection task are *color detecting* and *leave puck*. The *color detecting*, *homing*, *avoiding*, and *creeping* behaviors are also qualified to indicate the color of puck the robot has found. Control of the robot's drive motors was the basis for selecting the constituent members of this behavior space. When active, each of the behaviors has exclusive control of the motors, and together they account for all activity (or inactivity) of the motors for the duration of the task.

6.3.1 Validating the Approach

In order to validate our approach to regime detection, we show that: (1) regime changes do happen at different time scales, and (2) our algorithm using multiple AMMs can detect such changes, as brought about by shifts in large mine density. We compare results from two versions of the mine collection task that are identical except for the environmental setup. The hypothesis is that the decrease in environment size and the increase in clear puck (large mine) density in the second version pushes the regime change to a different time scale, most likely smaller.

The first version of the task uses an 11×14 foot (large) Corrall with 9 clear and 18 black pucks evenly distributed throughout (Figure 6.1: Left). With no *a priori* information about the

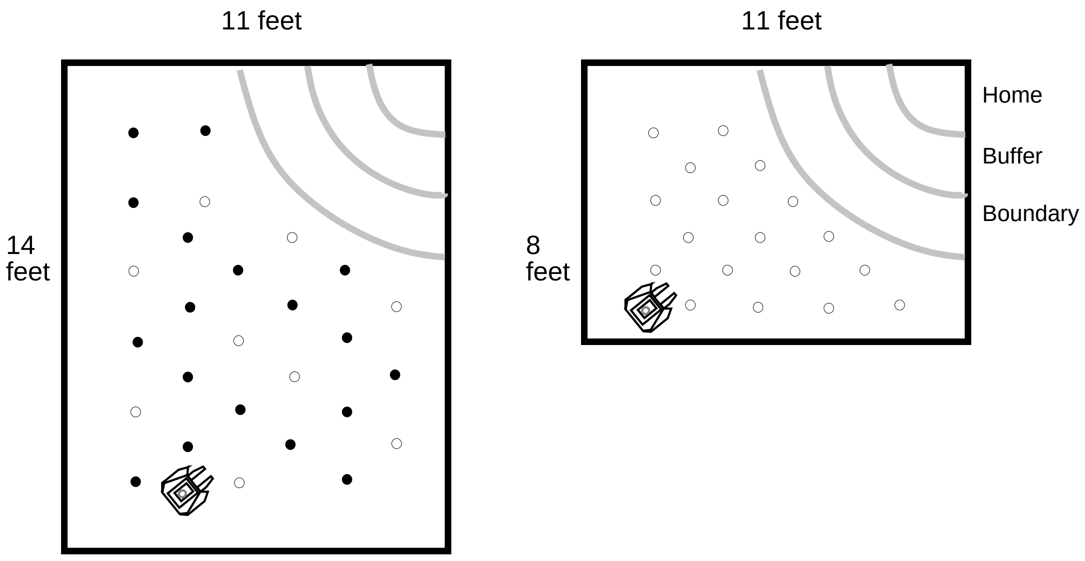


Figure 6.1: Two versions of the mine collection task environment: (Left) 11 $\times$ 14 foot Corrall with 9 clear and 18 black pucks; (Right) 11 $\times$ 8 foot Corrall with 18 clear pucks.

environment, the robot must collect only the clear pucks (i.e., large mines), while executing the regime detection algorithm to determine when to switch to black pucks (i.e., small mines). (In reality, data were sent via a serial radio link to an off-board Power Macintosh G3/266 which performed the regime detection algorithm and notified the robot of any regime changes. This was done because programming limitations of the R2's and the Behavior Language made implementing the algorithm on-board an R2 extremely difficult. These limitations are platform-specific.) In the second version, the Corrall is decreased in size to 11 $\times$ 8 feet (small) and only 18 clear pucks are used (Figure 6.1: Right). The key statistic of interest in these two versions is the time scale at which the robot detects a regime change and decides to begin collecting black pucks (small mines).

We complete the description of the validation experiment by presenting the parameter values used in the regime detection algorithm: the minimum time scale $\tau = 5$; the number of statistics kept for each model was $k = 8$; function $f_i = 2^i\tau$; Stat(m_i) returned the number of pucks that had been collected during the lifetime of AMM m_i; and DetectRegime(Q_i) performed an analysis of variance (ANOVA) on two groups of data (namely, the first and second $\frac{k}{2}$ values in Q_i), to determine if the means were different at a significance level of 10%, indicating a significant environmental shift. Since in each trial the robot was initialized to collect clear pucks (large mines), DetectRegime(Q_i) essentially determined if the number of clear pucks changed significantly enough over k consecutive intervals of size f_i to indicate a regime change.

We conducted five experimental trials in each of the two environments and gathered data about the time scale at which regime detection occurred. In each of the 10 trials, the algorithm successfully detected a regime switch. In the large Corrall environment, the mean time scale of

Puck type	Trial #				
	1	2	3	4	5
Clear pucks	4	8	15	10	14
Black pucks	8	4	16	9	10

Table 6.1: Pucks remaining in the environment at the end of each trial of the proportion maintaining mine collection task.

detection was 1024, while in the small Corrall it was 256. (Since data were collected at 2 Hz, this translates to approximately 512 seconds and 128 seconds, respectively.) A hypothesis test based on Student's t distribution (Freund 1992) indicates that the two means in the experiments are statistically different at a significance level of 1%. Thus, we have validated our approach by showing that regime changes do occur at different time scales (even in the same task but with different environments), and that our algorithm is able to detect such changes. Next, we describe a more sophisticated use of our approach.

6.3.2 Maintaining the Proportion of Mines

In a more complex version of the mine collection task, the robot is required to maintain the proportion of large to small mines in the environment at a specified value p. A significant switch in this value indicates a non-local regime switch, since p itself is a non-local measure. Once again, the robot begins by collecting large mines, but this time switches to small mines when the observed proportion p_{obs} is significantly different from p, and $p_{obs} < p$. Conversely, the robot switches back to large mines when $p_{obs} > p$ and this difference is significant. The goal of this experiment was to determine whether the robot could detect multiple consecutive regime changes in its environment due to shifts in the proportion of large to small mines.

For this experiment, the Corral was 11×8 feet and contained 18 each of clear and black pucks. The parameter values used in the regime detection algorithm were: $\tau = 5$; $k = 4$; $f(i) = 2^i\tau$; Stat(m_i) returned the proportion of clear to black pucks encountered; and DetectRegime(Q_i) performed an analysis of variance (ANOVA) at a significance level of 10% on Q_i and a list of length k having all values equal to p. The proportion p was set to 1.0, indicating that the robot should try to maintain equal numbers of the two types of mines. Whenever a regime switch was detected, the regime detection algorithm was re-initialized so as to be able to detect the next regime change.

In this experiment, a trial was considered complete when the robot detected two consecutive regime changes. The robot successfully did so in each of the five trials that were conducted. Table 6.1 shows the numbers of clear and black pucks remaining in the environment at the end of each trial. The correlation between the numbers is quite large ($\rho = 0.70$) and indicates that their proportion tended to be close to 1.0. Thus, not only was the robot able to detect multiple consecutive regime changes, but was also effective in maintaining the desired proportion of pucks (mines).

6.3.3 Maximizing Reward

An alternate experimental scenario requires the robot to maximize the expected reward garnered from collecting mines. Instead of designating *a priori* that the robot begin by collecting large mines (as in the previous experiments), the robot is told the reward value associated with each type of mine and must decide which to collect in order to maximize its total reward. Reward values can be set in proportion to a mine's explosive power, thus making reward maximization identical to minimizing the mine field's destructive potential. Regime detection enters the scenario when the environment is non-stationary, i.e., puck densities shift as they are collected or replaced. The following chapter describes this scenario in detail.

6.4 Summary

This chapter presented a novel approach that enables an agent to detect and respond to global environmental regime changes having no *a priori* knowledge or models of the environment, and limited to only local sensing. Multiple AMMs were constructed at different time scales and used to derive sets of statistics that were analyzed to detect a regime change. The approach was successfully validated on a physical mobile robot performing a land mine collection task. The next chapter presents another application in the non-stationary domain: reward maximization.

Chapter 7

AMMs in Non-Stationary Problem Domains: Reward Maximization

This chapter explores a second application of AMMs and behavior-based control to modeling agent-environment interaction dynamics in non-stationary problem domains. The problem explored in this chapter is reward maximization. Similar to the approach to regime detection in Chapter 6, the approach here also uses multiple AMMs to monitor the interaction dynamics at different time scales, but for the purposes of estimating the state of the environment. The approach is validated with a real mobile robot performing a mine collection task in both abruptly and gradually changing environments.

7.1 Introduction and Motivation

In certain classes of tasks, an agent may be required to perform optimally with respect to the information it possesses about the structure of its environment. Reward maximization may be used as a means of quantifying performance. In that framework, the agent receives reward (e.g., points) in proportion to its performance. Reward maximization in a non-stationary environment requires the agent to be able to estimate the state of the changing environment. There are a number of issues that can compound the difficulty of this problem:

- The agent may have no *a priori* knowledge of the environment and thus also lack a baseline for gauging the non-stationarity of the environment.
- Given only local sensing capabilities, the agent may require a significant amount of time to estimate the state of the environment. Any estimate of state, however, may be outdated in a non-stationary system.
- The nature of the task may be stochastic, with uncertainties large enough to preclude an effective predictive model of environmental state, or dynamics too complex to make the development of such a model feasible or tractable. Alternatively, however potentially simple the system, there may be no *a priori* data with which to instantiate a model.

- Further, in a stochastic system, the variability associated with performing a task (or elements there of) may be enormous and effectively mask gradual shifts in the environment. Conversely, in a system with very low variability, even minute shifts may be easily detected. Thus, effective estimation of environmental state requires an understanding of the system's variability (as often measured by variances, covariances, etc.).

- Depending on the task or environment, the time scale at which the non-stationarity manifests and thus can be detected may differ. For example, in one task, the environmental change may be almost instantaneous, detectable between one moment and the next. In another task, the change may be slow and incremental, requiring the examination of a large time interval for detection. Hard-coding the agent with a specific time scale to use for state estimation can be problematic. A time scale that is too small makes the agent incapable of detecting the change. Conversely, a time scale that is unnecessarily large increases the time required to detect the change and may be undesirable in time-critical situations.

As a concrete example, consider the task of collecting undetonated land mines in a field. Assume that there are two types of mines, large and small, with destructive power proportional to their size. The robot's goal is to minimize the destructive power of the mine field as much as possible during a given period of time. When the robot is given points in proportion to the destructive power of the mines it collects, the goal becomes equivalent to reward maximization. To accomplish its goal, the robot must have enough data about its environment (the field) to intelligently decide whether it is best to collect large mines or small ones at each point in time. The difficulty of this task is compounded when the issues mentioned above apply. The task is likely stochastic, with unknown variability. The robot may have no *a priori* information about the numbers of large and small mines in the field, their distributions, or relative proportions. The robot may also lack global sensing of the mines in the field. These limitations relegate the robot to estimating the environmental state while performing the task. With only an estimate, however, the robot may not perform in a globally optimal manner. The heart of this problem, therefore, is to use the best possible estimate of environmental state given the limitations of the system.

If the task environment is stationary, then all of the data the robot gathers may be used to estimate the state, with more data presumably providing a better estimate. Conversely, for non-stationary environmental state estimation, some mechanism must exist for discarding old data. This is a tricky proposition. If too much data are discarded, the estimate may be susceptible to noise and variance; if too little are discarded, the estimate may be skewed and not accurately represent the current state. This is analogous to the issues of overfitting and underfitting generally encountered in machine learning (Mitchell 1997). The appropriate amount of data to be kept is not necessarily static and pre-determinable, but rather, depends on the variances of the system and the type of non-stationarity exhibited. Low variances require less data (i.e., a smaller time scale) to characterize, as does non-stationarity exemplified by abrupt shifts. Both high variances and gradually shifting non-stationarity require greater amounts of data (i.e., a larger time scale) to characterize. Thus, a mechanism for estimating environmental state must accommodate both the variances and the type of non-stationarity exhibited by the system. Additionally, since multiple types of relevant non-stationarity and variances may exist in the system, a state estimation procedure capable of dynamically compensating is desirable.

We propose an algorithm that provides a moving average estimate of the state of a non-stationary system. The algorithm dynamically adjusts the window size used in the moving average to accommodate the variances and type of non-stationarity exhibited by the system, while discarding outdated and redundant data. Our focus is the application of the algorithm to the problem of reward maximization in a non-stationary environment. Similar to the algorithm in Chapter 6, the algorithm here also uses multiple AMMs to capture interaction dynamics at different time scales for evaluations at those time scales. The state of the environment is estimated indirectly though the robot's interaction with it. As task execution continues, AMMs are dynamically generated to accommodate the increasing time intervals. Sets of statistics from the models are used to determine whether old data and AMMs are redundant/outdated and can be discarded. This approach requires no *a priori* knowledge, uses only local sensing, and captures the notion of time scale. Additionally, it works naturally with stochastic task domains where variations between trials may change the most appropriate amount of data for state estimation.

In the next section, we use AMMs and the evaluations from Section 4.3 in our dynamic moving average algorithm.

7.2 Dynamic Moving Average Algorithm

To manage the amount of data used to estimate the state of a non-stationary environment, we present an algorithm which essentially computes a moving average with a dynamic window size. The algorithm maintains multiple AMMs for different time intervals, and uses a t-test and F-test on comparable values from the different AMMs to determine which AMM provides the best information. These tests allow the algorithm to adjust the window size of the moving average to accommodate both the amount of variance in the system and the type of non-stationarity (ranging from abrupt to very gradual). The window size of the moving average is allowed to grow by maintaining and expanding old AMMs, and is shrunk by deleting AMMs. We now present the algorithm.

```
1.  Let $\mathcal{L}$ be a queue-like list of AMMs, with $\mathcal{L}_0$ as the first element.
    Initialize $\mathcal{L}$ to contain one AMM.
2.  Let $\varpi_t$ and $\varpi_F$ be constants specifying the significance levels
    for the $t$-test and $F$-test, respectively.
3.  Let mode be a variable designating the two modes of the algorithm.
4.  For each new input symbol do the following:
5.      Update each AMM in $\mathcal{L}$ with the new input.
6.      If it is time to create a new AMM, then:
7.          Create a new AMM and add it to $\mathcal{L}$.
```

```
8.    Compute the mean first passage matrix for $\mathcal{L}_0$, extract the desired values
      and calculate their associated variance and degrees of freedom.
9.    Do the same for $\mathcal{L}_1$.
10.   Perform an $F$-test between the variances calculated for $\mathcal{L}_0$ and $\mathcal{L}_1$.
11.   If the significance level returned by the $F$-test is
      less than $\varpi_F$ (i.e., the variances are different),
      then set mode=1, else set mode=2.
12.   If mode==1, then let $i$ be the index of the first AMM in $\mathcal{L}$ after $\mathcal{L}_0$
      that has either significantly different variances or significantly
      different means (i.e., significance level $< \varpi_F, \varpi_t$).
      If such an $i$ exists, delete $\mathcal{L}_0$ through $\mathcal{L}_{i-2}$ and use the new
      $\mathcal{L}_0$ as the best estimate of the state.
13.   If mode==2, then let $i$ be the index of the first
      AMM in $\mathcal{L}$ after $\mathcal{L}_0$ that has neither significantly
      different variances nor means (i.e., significance
      levels $> \varpi_F, \varpi_t$). If such an $i$ exists,
      delete $\mathcal{L}_0$ through $\mathcal{L}_{i-1}$ and use the new $\mathcal{L}_0$ as
      the best estimate of the state.
14.   If no such $i$ exists for either value of mode, then
      do not delete any AMMs and use the current $\mathcal{L}_0$
      as the best estimate.
```

There are several characteristics of the algorithm worth noting. The decision criterion used to create a new AMM (line 6) is very general. For example, a new AMM might be created after a certain period of time, a certain number of input symbols, or when a particular input symbol is observed. In the experiments described below, a new AMM is created every time the robot finds an object (puck) to collect.

The algorithm adjusts the amount of data in the moving average to accommodate the variance in the system. This is accomplished by considering deletion of $\mathcal{L}_0$, the AMM representing the largest time window of data, only when its variance is comparable to (i.e., not significantly different from) that of another AMM. When the variability in the system is high, the AMMs require more data (i.e., larger windows) to acquire comparable variances. When variability is low, less data are required to accurately characterize the variances of the system.

The algorithm also has two distinct modes. In the first mode (*mode*=1), the algorithm removes redundant/old data. In systems with very gradual non-stationarity, this mode effectively maintains a good moving average estimate of the state. When there is an abrupt change in the system, the means and variances of adjacent AMMs may become increasingly different as more data are collected, causing the first mode to stall (i.e., not delete old AMMs). The second mode (*mode*=2) solves this problem by comparing non-adjacent AMMs to find two that are not significantly different. The second mode then "jumps over" the intervening AMMs by deleting them.

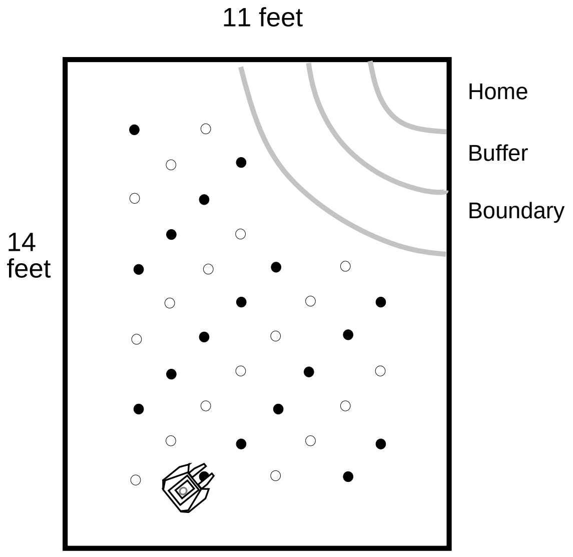


Figure 7.1: The mine collection task: setup for validation of the reward maximization criterion.

One final item of note is the importance of the ϖ_t and ϖ_F thresholds for significance level. Both values must in the interval $[0, 1.0]$, with a significance level of 0.05, or less, generally considered significant. The effect of ϖ_t and ϖ_F is to adjust the size of the moving average window. Extremely large values of both thresholds produce a very large window with excessive smoothing and a potentially skewed estimate of state. Very small values of the thresholds result in a small window and state estimation that is prone to overfitting. Empirical tests suggest that values in $[0.01, 0.1]$ tend to work fairly well in the algorithm, with relatively little sensitivity. It should also be noted that the experiments described later use this algorithm in real time.

The experimental verification of this chapter is done using the land mine collection task of Section 6.3. Figure 7.1 shows how the Corrall was setup. We now present the validation experiment and results for the reward maximization criterion.

7.3 Experiment 1: Validation of the Reward Maximization Criterion

Before demonstrating reward maximization in a non-stationary version of the mine collection task, we first validate the reward maximization criterion in a stationary environment. This validation is necessary to ensure that our subsequent results are not biased by an invalid assumption about the value of reward maximization. If it were shown that our reward maximization criterion did not improve performance over random behavior, then the utility of our dynamic moving average algorithm in the non-stationary version of the task would be suspect.

We now more formally define the reward maximization criterion. Let $\mathcal{R}_s$ and $\mathcal{R}_l$ be the rewards for small and large mines, respectively. Let τ_s^f be the expected time required for the robot to find a small mine, and let τ_s^d be the expected time to deliver the mine to the goal location once it has been found. Similarly, τ_l^f and τ_l^d represent these times for large mines. The robot maximizes its reward by deciding for each mine found whether to deliver it or leave it in search of a higher valued mine. The action chosen is the one that maximizes the expected reward per unit time (and thus the overall expected reward). If the robot finds a small mine, and the inequality

$$\frac{\mathcal{R}_s}{\tau_s^d} > \frac{\mathcal{R}_l}{\tau_l^f + \tau_l^d}$$

holds, then delivering the small mine maximizes reward. Otherwise, the small mine should be left and a large mine sought. The complementary inequality is used when a large mine is found.

The main issue in evaluating the inequality is calculating τ^d and τ^f for each mine type. One could maintain internal variables that record these values. Our approach, however, is to calculate these values from the robot's AMM. As discussed previously, each element, e_{ij}, of E gives the mean first passage from state i to state j. E also contains the values for τ^f and τ^d. τ^f is simply the entry in E associated with the minimum mean time from a *wandering* state to a *puck color* state, and τ^d is the minimum mean time from a *puck color* state to a *wandering* state. In other words, if s_1 is the input symbol for the *wandering* behavior and s_2 is the input symbol for the *puck color* behavior, then $\tau^d = \varepsilon_{21}$ and $\tau^f = \varepsilon_{12}$. Since these states are qualified by the color of puck the robot possesses, τ_s and τ_l can be distinguished.

We performed two sets of experiments in this validation: one control set where the robot collected both types of pucks without discrimination, and one set with reward maximization allowing the robot to decide which pucks to collect. For these experiments, reward values were set in proportion to a mine's explosive power, thus making reward maximization identical to minimizing the mine field's destructive potential. The setup for both sets of experiments was identical, with 18 clear pucks (large mines) and 18 black pucks (small mines) evenly distributed in the Corrall (Figure 7.1). Each clear puck had a reward of 4 points, while each black had a reward of 1 point. In addition, the environment was kept stationary by replenishing collected pucks. We performed five one-hour trials for each experiment. Using the reward maximizing criterion, the robot accrued an average of 46.6 points (standard deviation of 6.8), while without reward maximization the robot averaged 37 points (standard deviation of 5.0). A hypothesis test based on Student's t indicates that the means are different at a significance level of 5%, and validates our reward maximization criterion.

We now present experimental results applying the dynamic moving average algorithm to an abruptly changing non-stationary version of this task.

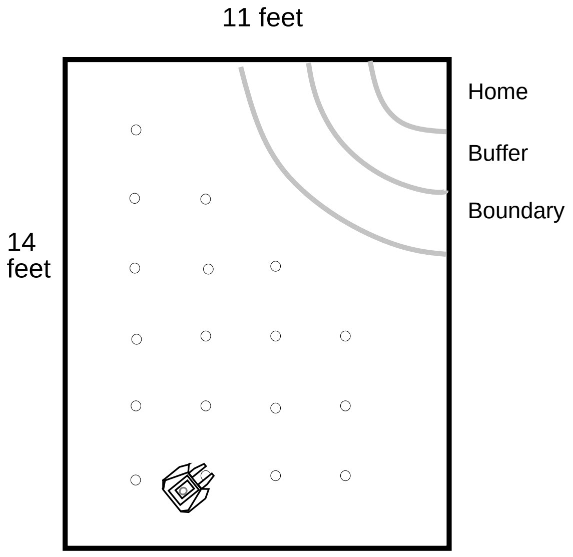


Figure 7.2: The mine collection task: setup for reward maximization in a non-stationary environment.

7.4 Experiment 2: Abruptly Changing Environment

In this set of experiments, we aim to show that, in an abruptly changing non-stationary version of the mine collection task, reward maximization with the addition of the dynamic moving average algorithm is superior to reward maximization alone. The hypothesis is that when the environment changes, average values of τ^f and τ^d become inaccurate and thus not effective for reward maximization. Using a dynamic moving average allows quicker adaptation to change.

The environment was first initialized to contain only 18 clear pucks (Figure 7.2). We ran one trial of approximately 20 minutes in this environment, during which the robot collected 4 clear pucks. This result is extremely variable: the actual time to collect 4 pucks could easily range between 10 and 30 minutes. In order to reduce the variability, the data from this one trial were used as a primer for all of the subsequent experiments, in which the white pucks (large mines) were replaced with black (small) ones. Doing so allowed us to focus on the key experimental parameter: the time required to adapt to the new environment. We considered the robot to have adapted to the new environment when it consistently began collecting black pucks.

We ran two experiments using reward maximization with the dynamic moving average algorithm and three experiments using reward maximization alone. In these experiments, the reward for a white puck was 7 points and the reward for a black puck was 1 point. For reward maximization alone, the mean time to adaptation was 47 minutes, while the mean time with the algorithm was 18.3 minutes. A t-test indicates that these means are different at a significance level of 0.01.

This strongly supports our hypothesis that the dynamic moving average algorithm allows quicker adaptation to abrupt non-stationarities.

7.5 Experiment 3: Gradually Shifting Environment

In this set of experiments, we test the effectiveness of the dynamic moving average algorithm for reward maximization in a gradually shifting environment. Instead of abruptly changing the color of the pucks, as in the previous experiment, here the environment shifts slowly as the robot collects pucks. Due to the high degree of variance in the mine collection task using physical robots, we anticipated that the number of experiments we would have to conduct in order to obtain statistical significance in the gradually shifting environment would pose a practical impossibility. The experiments described in this section were therefore conducted in a simulation of the mine collection task.

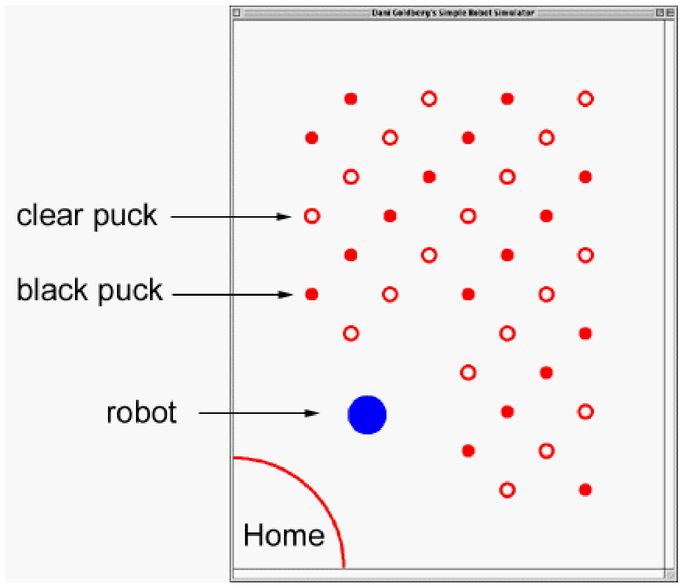


Figure 7.3: The simulated mine collection task: setup for reward maximization in a gradually shifting non-stationary environment.

The simulation was initialized to contain 18 clear pucks worth 10 points each, and 18 black pucks worth 1 point each (Figure 7.3). Three experimental scenarios were examined:

1. **random**: the robot collects any puck it encounters regardless of the color.
2. **control**: the robot uses the reward maximization criterion, but does not employ the dynamic moving average algorithm used to compensate for non-stationarity.
3. **algorithm**: the robot uses both reward maximization and the dynamic moving average algorithm for state estimation.

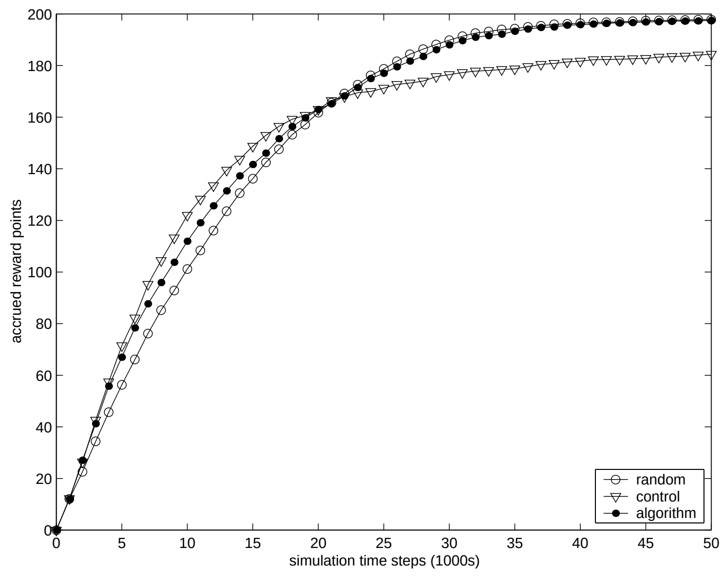


Figure 7.4: Average accrued reward for the three experimental scenarios (puck point values: black=1, clear=10).

	random	**control**	**algorithm**
Expected average reward	151.6	149.3	153.8
Standard deviation	5.7	5.7	5.5

Table 7.1: The average reward points the robot is expected to have accrued during the **random**, **control** and **algorithm** scenarios (puck point values: black=1, clear=10).

We conducted trials of 50,000 simulation steps for each scenario: 200 trials of the **random** scenario, 40 of **control**, and 100 of **algorithm**, with the actual number determined by the desired level of statistical significance. The data gathered included the time at which each puck was collected, allowing us to calculate the accrued number of reward points. Figure 7.4 presents the average number of reward points accrued at 1000-time-step intervals for each of the three scenarios. The maximum possible accrued reward is 198 points, corresponding to the collection of all 36 pucks. Both the **random** and **algorithm** scenarios essentially reach this maximum by the end of each trial (with average accrued points of 197.8 and 197.5, respectively). The **control** scenario outperforms the other two until around 20,000 time steps, then quickly declines ending with an average reward of 184.4. The discrepancy between the **algorithm** and **control** cases illustrates the importance of eliminating outdated information, and the effectiveness of our algorithm in doing so. As a further comparison, we calculate the number of reward points that the robot is expected to have accrued on average during the course of a trial: 151.6 for **random**, 149.3 for **control**, and 153.8 for **algorithm** (Table 7.1). The pair-wise comparison of the data using a two-sample version of Student's t test indicates significantly different means at p-value< 0.02. The superiority of the **algorithm** case illustrates the effectiveness of our moving average algorithm for reward maximization in a gradually shifting environment.

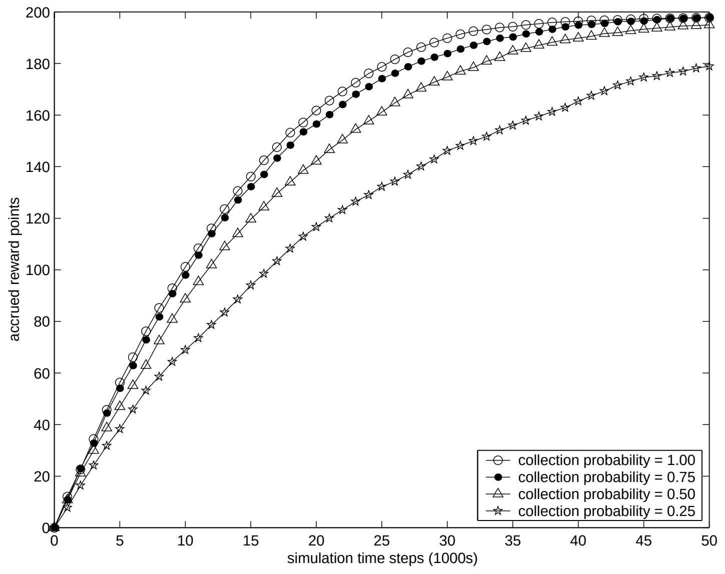


Figure 7.5: Average accrued reward for the four versions of the random scenario with different probabilities for collecting pucks (puck point values: black=1, clear=10).

Probability of collection	1.0	0.75	0.50	0.25
Expected average reward	151.6	148.5	140.3	118.3
Standard deviation	5.7	6.2	7.3	10.9

Table 7.2: The average reward points the robot is expected to have accrued during four versions of the **random** scenario with different collection probabilities (puck point values: black=1, clear=10).

It should be noted that the **random** scenario used above lies along a continuum of possible experiments distinguished by the probability with which the robot collects (or discards) each puck it encounters. For comparison with the **control** and **algorithm** scenarios, we used the random case in which the robot collects 100% of the pucks it encounters. Intuitively, one would expect this to be the best performing of the random cases. When the probability of collecting a found puck is less than 1.0, the robot wastes more time searching for pucks but does not improve its reward since it discards both high-valued and low-valued pucks with equal probability. To verify our intuition, we conducted 80 trials for each of three lower probabilities of collection: 0.75, 0.50, 0.25. As expected, the data show that there is a continual significant decrease in accrued reward as the probability of collection decreases. The expected reward points accrued on average during the course of a trial (in order of decreasing collection probability) are: 151.6, 148.5, 140.3, 118.3 (Table 7.2). Pairwise *t*-tests indicate that the four random scenarios are indeed different at a significance level of 0.0001, and Figure 7.5 visually demonstrates this difference. Thus, it might initially seem that the random cases with lower collection probabilities are more appropriate for comparison with the **algorithm** and **control** scenarios. It is, however, the collection probability of 1.0 that is the most difficult challenge for our algorithm, and consequently the one used for the comparisons described here as well as those in Section 7.3.

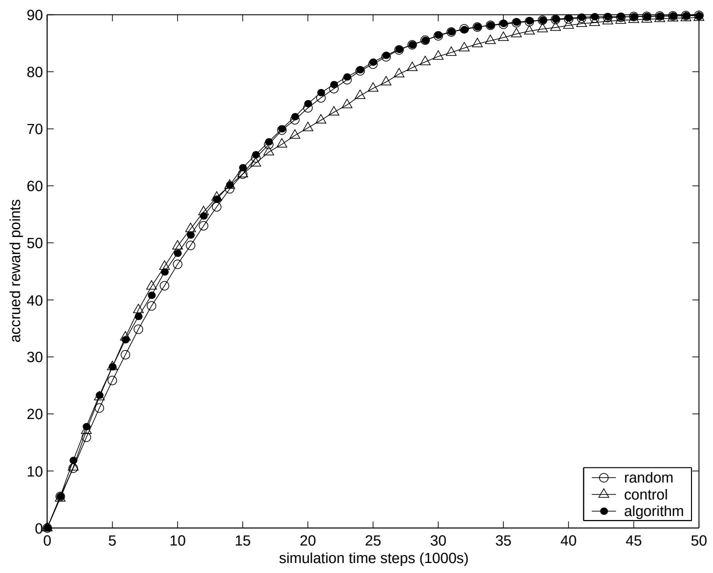


Figure 7.6: Average accrued reward for the three experimental scenarios with close point values for pucks (black=1, clear=4).

	random	**control**	**algorithm**
Expected average reward	69.02	67.99	69.67
Standard deviation	2.28	2.16	2.46

Table 7.3: The average reward points the robot is expected to have accrued during the **random**, **control** and **algorithm** scenarios with close puck point values (black=1, clear=4).

We also tested our algorithm in a more challenging set of experiments where the point values of pucks were closer together (black=1, clear=4). Close point values make the difference in reward for collecting the "wrong" versus the "right" colored puck very small, and thus further sensitize the performance of the robot to the accuracy of the estimate of environmental state. In these experiments, we compared data from 140 trials of the **control** and **algorithm** scenarios, and 200 trials of the **random** scenario (all trials being run to $50,000$ simulation steps). The expected reward accrued on average during these scenarios was, respectively: 67.99, 69.67, 69.02 (Table 7.3). A *t*-test shows these results to be different at a significance level of 0.02. The superior performance of the **algorithm** scenario given the closeness of the data (Figure 7.6) helps illustrate the effectiveness of our AMM-based, moving-average state estimation algorithm using dynamic windowing.

7.6 Summary

This chapter explored the use of AMMs and behavior-based control for modeling interaction dynamics in a non-stationary environment. Multiple AMMs were used as part of a moving average algorithm with dynamic windowing, which was applied to estimating the environmental state. This estimate provided a robot performing the land mine collection task with the information to make performance-improving decisions about which type of mine to collect.

Chapter 8

Parametric versus Nonparametric AMMs

This chapter compares the effectiveness of the erstwhile used *parametric* version of AMMs (assuming normally distributed state durations) to a *nonparametric* alternative (using the raw duration data without fitting it to a parametric distribution). The motivation for this empirical comparison is the observation that the real-world, mobile-robot, behavior-execution data modeled in the previous chapters with parametric AMMs is, in fact, non-normal. The question is how the violation of normality impacts the effectiveness of parametric AMMs. The answer, as we will see, is that the violation makes parametric AMMs less effective than their nonparametric counterpart when the order of the system is unknown.

8.1 Introduction

In this chapter, we examine the effectiveness of modeling when the data used violate underlying assumptions of the model. The impact of such a violation on the desired application is not necessarily obvious. Perhaps the combination of the modeling approach, the training data, and the target application is fairly insensitive to the deviation, which therefore has negligible impact on the performance of the system. Alternatively, this combination of factors might be quite sensitive to the deviation in assumptions, leading to relatively poor performance. Given the potentially complex interaction of factors involved, an empirical study is a practical option for determining the relative merits of competing approaches to modeling. This chapter presents such an empirical study, focusing specifically on a comparison between parametric (Gaussian/normal) and nonparametric versions of AMMs as applied to the generally non-normal robot data from the foraging task (Chapter 2). The question answered in this chapter is: *Might the results of the previous chapters have been different if a nonparametric version of AMMs were used instead of the parametric version?*

The assumption of normally distributed data is not uncommon in machine learning and statistics. In our case, this assumption led to a more parsimonious AMM representation by allowing the data to be incrementally summarized in mean and variance values. The nonparametric version, by contrast, must store all of the data at multiple Markovian orders. The use of parametric statistics also simplified certain expectation calculations (Chapter 4). It is intuitively apparent, however, that the robot foraging data violate this assumption, simply by noting that a robot can not spend negative time in a state.

Figure 8.1 shows the actual distribution of data from the execution of four behaviors. It is clear, both graphically and from the chi-square goodness-of-fit test (Freund 1992, pp. 487–489), that this violation is quite severe, and that the robot data do not nicely fit any standard parametric distribution. While this result does not invalidate the work in previous chapters, it does lead us to question whether the results might have been better had we not used a parametric (Gaussian) version of AMMs, and instead used a nonparametric version making as few distribution assumptions as possible. In order to answer this question, we implemented a nonparametric version of AMMs. To the author's knowledge, this is the first incremental, higher-order, nonparametric, SMP-like model. Details of the model representation and incremental construction algorithm are found in Appendix B. A review of relevant parametric and nonparametric statistics literature is provided in Sections 3.3 and 3.4. The next section describes one of the key differences between parametric and nonparametric AMMs — the test for node-splitting.

8.2 Parametric and Nonparametric Node-Splitting

There are two types of node-splitting that can occur in the AMM construction algorithm: one is due to link traversal inconsistencies; the other occurs if the duration in a state differs significantly depending upon the particular multi-link transition sequence that enters that state. Whereas the former type of node-splitting relies on the binomial distribution, the latter depends on the SMP-like distribution that characterizes the time spent in a state. Node-splitting based on durations, therefore, encompasses the distinguishing characteristics of parametric and nonparametric AMM construction. It will also be the primary cause of differences between parametric and nonparametric AMMs in the forthcoming evaluation study.

In parametric AMMs, a t test (based on Student's t distribution) is used to determine a significant deviation in the mean time spent in a state. The particular form of the test allows for non-identically distributed populations (Press et al. 1992). The key point regarding the t test is that it assumes that observations are independent and are drawn from normally distributed populations. When these assumptions hold, the t test is a powerful test of location (Siegel & Castellan 1988). It also tends to be sensitive, however, to outliers and deviations from its assumptions.

In nonparametric AMMs, the hypothesis test used to determine a significant discrepancy in state duration is a median test by Fligner & Rust (1982). This test makes few distribution assumptions, allowing the underlying distributions to be both unequal (i.e., of different shapes) as well as asymmetric. This test, however, can have difficulties if the distributions are severely asymmetric with many values equal to the median. Section 3.4.1 presents the details of the test and

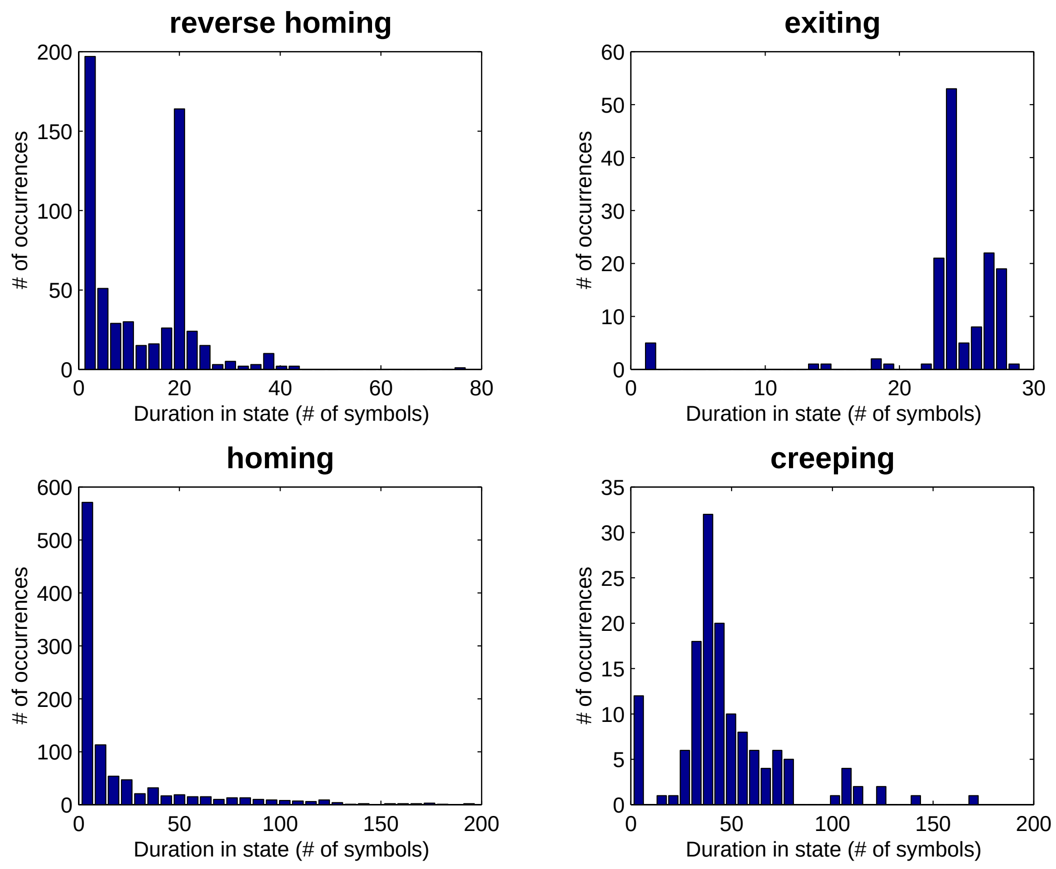


Figure 8.1: The distributions associated with the execution of four behaviors (*reverse homing*, *exiting*, *homing*, *creeping*) in the foraging controller (Section 2.3). The graphs were generated with real data captured from the physical mobile robots.

Appendix C provides tables of critical points. The next section describes the simulation used in evaluating parametric and nonparametric AMMs.

8.3 Simulation and Evaluation

In order to evaluate the relative merits of parametric and nonparametric AMMs, we use a simulation of the foraging task employing real robot data collected from seven behaviors: *avoiding*, *wandering*, *puck detecting*, *homing*, *creeping*, *exiting*, and *reverse homing* (Section 2.3). When active, each of these behaviors has exclusive control of the motors, and together they account for all activity (or inactivity) of the motors during the foraging task, i.e., they constitute a *behavior space* for the task (Section 4.4). To gather the data used in the simulation, we polled the behavior activity of a robot at approximately 2 Hertz, as it performed the foraging task over a period of 10 hours.

There are two reasons why a simulation, rather than experimentation on real robots, is used in this study. First, since we designed the simulation, we know the precise characteristics of the system that the AMMs attempt to model. This provides exact ("theoretical") baseline data with which to evaluate the performance of the AMMs. Due to the possibility of hidden state, complex stochasticity, and non-stationarity, it would be difficult to derive any such exact system information in the real world. Second, even if sufficiently faithful baseline information about the real world system were available, the hundreds of hours of trials that would be necessary with real robots would still pose a practical impossibility.

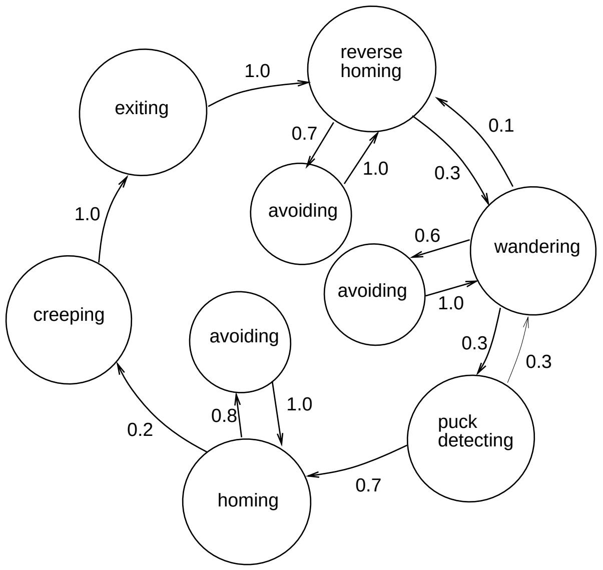


Figure 8.2: The transition model of the foraging task used by the simulation for evaluating parametric and nonparametric AMMs.

Figure 8.2 presents the transition diagram of the foraging task used by the simulation. During each step of the simulation, a transition is made from the current state to a new state according to the probabilities in the graph. In the new state, the simulation randomly samples a value from the population of durations for that behavior, from the real robot data. For example, upon entering *wandering*, the simulation might pick a value of 9 from the distribution of durations that the real robot spent in the *wandering* behavior. The simulation then feeds 9 symbols representing the *wandering* behavior to the parametric or nonparametric AMM construction algorithm. Note that the simulation is second-order Markovian, since when transitioning from the *avoiding* behavior, the system must remember the previous behavior in order to be able to return to it. In other words, a transition from the *avoiding* behavior not only depends on the current state, but also the previous one.

Since the simulation has the actual sequence of state transitions that were made, and the duration spent in each state, it is possible to compute the "theoretical" value for the mean first passage between any two states. This can be compared to the corresponding values derived from the AMMs using Markov chain expectation calculations. Our interest is in the mean first passage between two states that represent the execution of particular behaviors, but because multiple states might represent these behaviors, we wish to calculate ε_{ij}, the *minimum mean first passage between two input symbols, i and j* (Section 4.3.1). This is one of the key AMM-based evaluations used in the previous chapters, and in particular, Chapter 7. Because there are 7 behaviors (input symbols) in the simulation, there are 49 distinct ε_{ij}. The evaluation metric, Δ_ε, for the performance of the parametric and nonparametric AMM implementations relies on the calculation of the sum of the absolute differences between corresponding values of ε_{ij}, i.e.,

$$\Delta_\varepsilon = \sum_{\text{all } i,j} \mid \varepsilon_{ij} - \widehat{\varepsilon_{ij}} \mid,$$

where ε_{ij} is the actual ("theoretical baseline") value from the simulation and $\widehat{\varepsilon_{ij}}$ is the value calculated from the AMM. Thus, smaller values of Δ_ε signify better performance by the AMM. The next section presents the simulation results.

8.4 Experimental Results

This section presents results of the study comparing the relative effectiveness of parametric and nonparametric AMMs in modeling the simulated version of the foraging task using real robot data. There are two main factors that influence the performance of the AMM construction algorithm, and which we explore in conjunction with the parametric/nonparametric distinction. One factor is the significance level of the binomial confidence interval test and the location test that are used to determine whether node splitting is required. The higher the significance level, the less confidence there is in the decision to split a node, and thus the more likely it is that the algorithm will incorrectly do so. The second factor is the user-specified maximum order of the AMM, $n_{\max}$. Each time the algorithm determines whether node splitting is necessary for the current state, it checks $\mathbf{T}^1, \ldots, \mathbf{T}^{n_{\max}-1}$, the $n_{\max} - 1$ structures containing statistics on multi-link transitions. Thus, the larger $n_{\max}$ is, the more opportunities the algorithm has (and, thus, the more likely it is) to incorrectly split a node. It is these incorrect splits that are the primary cause of poor performance.

We first consider a node-splitting significance level of 0.05 and $n_{\max} = 10$. Figure 8.3 shows the average performance over 100 trials of 10000 simulation steps, using the Δ_ε metric described previously. The performance of the parametric version degrades after 4000 time steps, indicating that it does not converge to a stable topology. The data clearly show that nonparametric AMMs model the simulation more faithfully than do parametric ones. This result is not very surprising considering that the non-normal robot data places the t test of the parametric AMM algorithm at a distinct disadvantage to the more robust median test of the nonparametric version. Figure 8.4 shows that this disadvantage is still evident at a node-splitting significance level of 0.01, although the performance of the parametric version is not as deficient as at a level of 0.05. Further simulation results at a node-splitting significance level of 0.005 support the continued superiority of the nonparametric version.

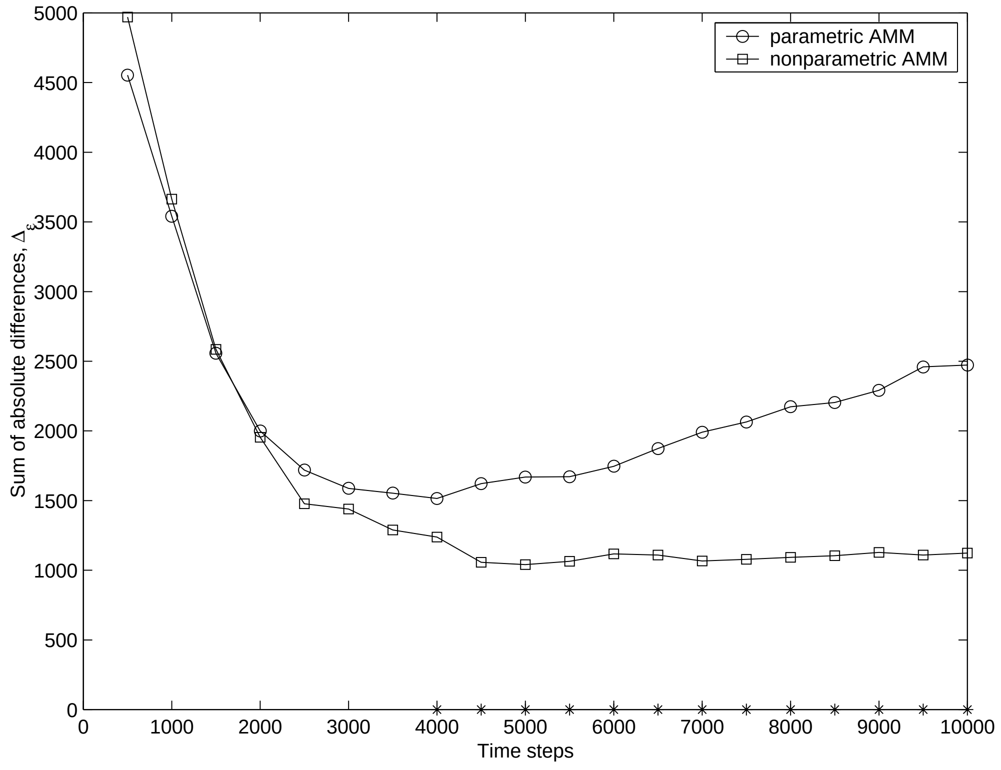


Figure 8.3: The performance, Δ_ε, of parametric and nonparametric AMMs with a node-splitting significance of 0.05 and $n_{\max} = 10$. This graph shows the average over 100 trials of 10000 simulation steps. Asterisks indicate a significant difference at a level of 0.01.

We now consider the impact of the user-specified maximum order of the model, $n_{\max}$. In general, we observe that, for both parametric and nonparametric AMMs, the value of $n_{\max}$ should be greater than or equal to the actual order of the system being modeled. This allows the model construction algorithm the chance to capture the correct order of the system. One caveat, however, is that $n_{\max}$ should also be close to actual order of the system. If, for example, the system is sixth order, it is better to set $n_{\max}$ to 8 than to 10. Of course, the user may not have a good idea of the order of the system, making it difficult to pick a good $n_{\max}$. Unfortunately, an unnecessarily high value of $n_{\max}$ increases the possibility of node-splitting errors at high orders, which can significantly impact the effectiveness of the model.

In support of these observations is the following experimental result: when $n_{\max}$ is set to 2, equaling the order of the simulation, there is no significant difference between the performance of parametric and nonparametric AMMs, regardless of the node-splitting significance level. The t test,

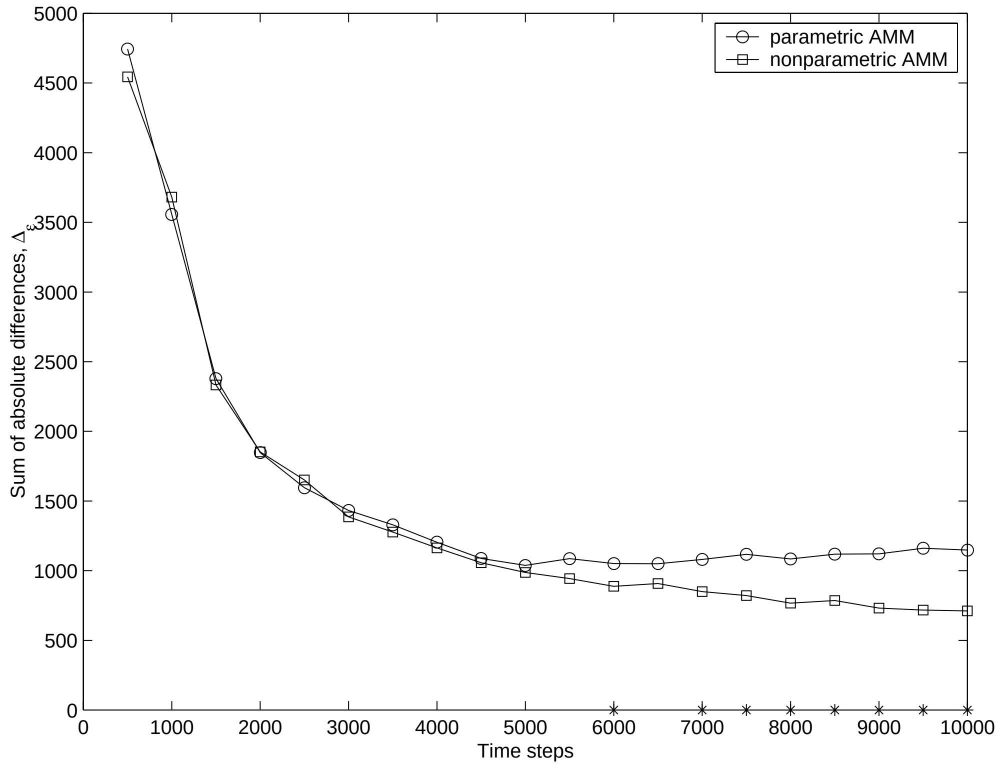


Figure 8.4: The performance, Δ_ε, of parametric and nonparametric AMMs with a node-splitting significance of 0.01 and $n_{\max} = 10$. This graph shows the average over 100 trials of 10000 simulation steps. Asterisks indicate a significant difference at a level of 0.01.

used in the parametric version, is particularly sensitive to violations of its assumptions, so when the parametric algorithm is limited to a low $n_{\max}$, it is also limited in the number of node-splitting mistakes it can make.

These results suggest that, in the previous chapters, the nonparametric version of AMMs might have been more effective in modeling the robot-environment interaction dynamics if we had had no idea of the order of the system and had to pick a large $n_{\max}$. Fortunately, we have extensive experience with the interaction dynamics arising in the variations of the foraging task, and felt confident that they were second order, and not higher. Thus, $n_{\max}$ was set to 2. It is therefore unlikely that the nonparametric implementation would have afforded us better performance in the applications of the previous chapters.

8.5 Summary

This chapter compared the effectiveness of the parametric AMMs (used in the previous chapters) to a nonparametric version making few distribution assumptions about state durations. The results of a simulation study showed that the nonparametric version of AMMs provides more faithful models when the data are not normally distributed and the user-specified order of the system is larger than the actual order. In the applications of the previous chapters, the user-specified order was 2, making it unlikely that nonparametric AMMs would have provided better performance.

Chapter 9

Summary and Future Directions

This dissertation presented a novel approach for capturing and evaluating, on-line and in real-time, the interaction dynamics between an agent and its environment. The approach is based on the synergistic combination of *augmented Markov models* (AMMs) and *behavior-based control* (BBC).

Augmented Markov models, a contribution of this dissertation, provide a compromise between the generality of semi-Markov processes and the computational simplicity of Markov chains. AMMs allow standard expectation calculations from Markov chain theory to be combined easily with popular statistical hypothesis tests (such as the t and F tests) that assume normal distributions, or their nonparametric counterparts. This dissertation presented an incremental AMM construction algorithm that dynamically restructures models to represent, in first-order form, non-first-order Markovian systems.

AMMs were used with behavior-based control to capture the execution history arising from the interaction between an agent performing a task and its environment. The use of behavior-based control, encompassing and abstracting both sensing and action, provides the representational expressiveness and parsimony necessary for on-line, real-time modeling with AMMs. The combination of AMMs and behavior-based control enables the effective evaluation of interaction dynamics and may be used to suggest application-dependent, performance-improving modifications to an agent's policy.

The effectiveness of the AMM-BBC approach was verified in both stationary and non-stationary mobile robot problem domains. Experimental results were provided for three applications in the stationary domain (fault detection, affiliation determination, dynamic leader selection) and two applications in the non-stationary domain (regime detection, reward maximization), all using variations of a foraging task. The results support the *Thesis* of this dissertation as presented in Chapter 1: *AMMs, in conjunction with behavior-based control, enable effective evaluation of agent-environment interaction dynamics and facilitate performance-improving solutions to application challenges.*

Many extensions exist to the work in this dissertation. Several possible ones are:

- *Node-merging*: This is a complement to node-splitting that would allow incorrect splits to be fixed when sufficient data are available to indicate the error.

- *Multiple simultaneous applications*: There is no reason why the same AMMs could not be used simultaneous for several non-conflicting applications. One possible example is fault detection concurrent with dynamic leader selection.
- *Heterogeneous groups*: It would be interesting to see how AMMs could be used to coordinate group activity among heterogeneous agents with highly disparate characteristics and capabilities.

Perhaps these, and others, will see the light of day.

Reference List

Anscombe, F. J. (1950), 'Table of the Hyperbolic Transformation $\sinh^{-1}\sqrt{x}$', *Journal of the Royal Statistical Society* **113**(2), 228–229.

Arkin, R. C. (1998), *Behavior-Based Robotics*, The MIT Press: Cambridge, Massachusetts.

Arkin, R. C. & Ali, K. S. (1994), Reactive and Telerobotic Control in Multi-Agent Systems, *in* 'From Animals to Animats: International Conference on Simulation of Adaptive Behavior', MIT Press, pp. 473–478.

Arkin, R. C. & Hobbs, J. D. (1993), Communication and Social Organization in Multi-Agent Systems, *in* 'From Animals to Animats: International Conference on Simulation of Adaptive Behavior', MIT Press, pp. 486–493.

Arkin, R. C., Balch, T. & Nitz, E. (1993), Communication of Behavioral State in Multi-Agent Retrieval Tasks, *in* 'IEEE International Conference on Robotics and Automation', IEEE Computer Society Press, pp. 588–594.

Bailey, B. J. R. (1981), 'Alternatives to Hastings' Approximation to the Inverse of the Normal Cumulative Distribution Function', *Applied Statistics* **30**(3), 275–276.

Bajcsy, R. (1988), 'Active Perception', *Proceedings of the IEEE* **76**(8), 996–1005.

Balch, T. (1997), Social Entropy: a New Metric for Learning Multi-robot Teams, *in* 'Proceedings of the 10th International Florida Artificial Intelligence Research Society Conference (FLAIRS-97)', AAAI Press, Daytona Beach, FL.

Balch, T. (2000), 'Hierarchical Social Entropy: An Information Theoretic Measure of Robot Group Diversity', *Autonomous Robots* **8**, 209–237.

Ballard, D. H. (1991), 'Animate Vision', *Artificial Intelligence* **48**(1), 57–86.

Beckers, R., Holland, O. & Deneubourg, J. (1994), From Local Actions to Global Tasks: Stigmergy and Collective Robotics, *in* 'Artificial Life IV, Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems', MIT Press, pp. 181–189.

Beer, R. D. (1993), 'A Dynamical Systems Perspective on Agent-Environment Interaction', *Artificial Intelligence* **72**, 173–215.

Blyth, C. R. (1986), 'Approximate Binomial Confidence Limits', *Journal of the American Statistical Association* **81**(395), 843–855.

Blyth, C. R. & Still, H. A. (1983), 'Binomial Confidence Intervals', *Journal of the American Statistical Association* **78**(381), 108–116.

Boutilier, C., Dean, T. & Hanks, S. (1999), 'Decision Theoretic Planning: Structural Assumptions and Computational Leverage', *Journal of Artificial Intelligence Research* **11**, 1–94.

Bradtke, S. J. & Duff, M. O. (1995), Reinforcement Learning Methods for Continuous-Time Markov Decision Problems, *in* G. Tesauro, D. Touretzky & T. Leen, eds, 'Advances in Neural Information Processing Systems', Vol. 7, The MIT Press, pp. 393–400.

Brooks, R. A. (1986), 'A Robust Layered Control System for a Mobile Robot', *IEEE Journal of Robotics and Automation* **RA-2**(1), 14–23.

Brooks, R. A. (1990), The Behavior Language; User's Guide, Technical Report AIM-1227, MIT AI Lab.

Brooks, R. A. (1991), Intelligence Without Reason, *in* 'Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)', Morgan Kaufmann, pp. 569–590.

Camp, B. H. (1951), 'Approximation to the Point Binomial', *Annals of Mathematical Statistics* **22**(1), 130–131.

Cao, Y. U., Fukunaga, A. S. & Kahng, A. B. (1997), 'Cooperative Mobile Robotics: Antecedents and Directions', *Autonomous Robots* **4**, 1–23.

Cassandra, A. R., Kaelbling, L. P. & Littman, M. L. (1994), Acting Optimally in Partially Observable Stochastic Domains, *in* 'Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-94)', Seattle, WA, pp. 1023–1028.

Chrisman, L. (1992), Reinforcement Learning with Perceptual Aliasing: The Perceptual Distinctions Approach, *in* W. Swartout, ed., 'Proceedings of the 10th National Conference on Artificial Intelligence', MIT Press, pp. 183–188.

Chu, J. T. (1956), 'Errors in Normal Approximations to the t, τ, and Similar Types of Distribution', *Annals of Mathematical Statistics* **27**(3), 780–789.

Cormen, T. H., Leiserson, C. E. & Rivest, R. L. (1990), *Introduction to Algorithms*, McGraw-Hill Book Company.

Drogoul, A. & Ferber, J. (1992), From Tom Thumb to the Dockers: Some Experiments with Foraging Robots, *in* 'From Animals to Animats II', The MIT Press: Cambridge, Massachusetts, pp. 451–459.

Fenstad, G. U. (1983), 'A Comparison Between the U and V Tests in the Behrens-Fisher Problem', *Biometrika* **70**(1), 300–302.

Fisher, R. A. & Cornish, E. A. (1960), 'The Percentile Points of Distributions Having Known Cumulants', *Technometrics* **2**, 209–225.

Fligner, M. A. & Policello, G. E. (1981), 'Robust Rank Procedures for the Behrens-Fisher Problem', *Journal of the American Statistical Association* **76**(373), 162–168.

Fligner, M. A. & Rust, S. W. (1982), 'A Modification of Mood's Median Test for the Generalized Behrens-Fisher Problem', *Biometrika* **69**(1), 221–226.

Fontán, M. S. & Matarić, M. J. (1998), 'Territorial Multi-Robot Task Division', *IEEE Transactions on Robotics and Automation* **14**(5), 815–822.

Freund, J. E. (1992), *Mathematical Statistics*, fifth edn, Prentice Hall.

Gat, E. (1998), On Three-Layer Architectures, *in* D. Kortenkamp, R. P. Bonnasso & R. Murphy, eds, 'Artificial Intelligence and Mobile Robotics: Case Studies of Successful Robot Systems', AAAI Press, pp. 195–210.

Gentleman, W. M. & Jenkins, M. A. (1968), 'An Approximation for Student's t-Distribution', *Biometrika* **55**(3), 571–572.

Ghosh, B. K. (1979), 'A Comparison of Some Approximate Confidence Intervals for the Binomial Parameter', *Journal of the American Statistical Association* **74**(368), 894–900.

Goldberg, D. & Matarić, M. J. (1997), Interference as a Tool for Designing and Evaluating Multi-Robot Controllers, *in* 'Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)', AAAI Press, Providence, Rhode Island, pp. 637–642.

Goldsmith, S. Y., Feddema, J. T. & Robinett, R. D. (1998), Analysis of Decentralized Variable Structure Control for Collective Search by Mobile Robots, *in* 'Sensor Fusion and Decentralized Control in Robotic Systems', Vol. 3523 of *SPIE Proceedings*, SPIE, Boston, Massachusetts, pp. 40–47.

Gordon, D. M. (1996), 'The organization of work in social insect colonies', *Nature* **380**, 121–124.

Hamaker, H. C. (1978), 'Approximating the Cumulative Normal Distribution and its Inverse', *Applied Statistics* **27**(1), 76–77.

Han, K. & Veloso, M. (1999), Automated Robot Behavior Recognition Applied to Robotic Soccer, *in* 'Proceedings of the IJCAI-99 Workshop on Team Behaviour and Plan Recognition', Stockholm, Sweden.

Hanson, S. J. (1990), Meiosis Networks, *in* D. S. Touretzky, ed., 'Advances in Neural Information Processing Systems 2', Morgan Kaufmann, San Mateo, CA, pp. 533–541.

Hasegawa, Y., Ito, Y. & Fukuda, T. (2000), Behavior Coordination and its Modification on Brachiation-type Mobile Robot, *in* 'Proceedings of the 2000 IEEE International Conference on Robotics and Automation', IEEE, San Francisco, CA, pp. 3984–3989.

Hawkes, A. G. (1982), 'Approximating the Normal Tail', *Statistician* **31**(3), 231–236.

Hettmansperger, T. P. & Malin, J. S. (1975), 'A Modified Mood's Test for Location with no Shape Assumptions on the Underlying Distributions', *Biometrika* **62**(2), 527–529.

Hettmansperger, T. P. & McKean, J. W. (1998), *Robust Nonparametric Statistical Methods*, Vol. 5 of *Kendall's Library of Statistics*, John Wiley & Sons.

Holland, O. & Melhuish, C. (2000), 'Stigmergy, Self-Organization, and Sorting in Collective Robotics', *Artificial Life* **5**(2), 173–202.

Johnson, N. L., Kotz, S. & Kemp, A. W. (1992), *Univariate Discrete Distributions*, Wiley Series in Probability and Mathematical Statistics, second edn, John Wiley and Sons.

Kaelbling, L. P., Littman, M. L. & Moore, A. W. (1996), 'Reinforcement Learning: A Survey', *Journal of Artificial Intelligence Research* **4**, 237–285.

Kemeny, J. G., Snell, J. L. & Knapp, A. W. (1966), *Denumerable Markov Chains*, D. Van Nostrand Company, Inc.

Koenig, S. & Simmons, R. G. (1996), Unsupervised Learning of Probabilistic Models for Robot Navigation, *in* 'Proceedings of the IEEE International Conference on Robotics and Automation', Vol. 3, pp. 2301–2308.

Košecká, J. & Bajcsy, R. (1993), Discrete Event Systems for Autonomous Mobile Agents, *in* 'Proceedings of the Symposium on Intelligent Robotic Systems', pp. 21–31.

Lew, R. A. (1981), 'An Approximation to the Cumulative Normal Distribution with Simple Coefficients', *Applied Statistics* **30**(3), 299–301.

Lin, J.-T. (1988), 'Alternatives to Hamaker's Approximations to the Cumulative Normal Distribution and its Inverse', *Statistician* **37**(4/5), 413–414.

Lin, J.-T. (1990), 'A Simpler Logistic Approximation to the Normal Tail Probability and its Inverse', *Applied Statistics* **39**(2), 255–257.

Lindström, M., Orebäck, A. & Christensen, H. I. (2000), BERRA: A Research Architecture for Service Robots, *in* 'Proceedings of the 2000 IEEE International Conference on Robotics and Automation', IEEE, San Francisco, CA, pp. 3278–3283.

Ling, R. F. (1978), 'A Study of the Accuracy of Some Approximations for t, χ^2, and F Tail Probabilities', *Journal of the American Statistical Association* **73**(362), 274–283.

Lund, H. H. & Pagliarini, L. (2000), RoboCup Jr. with LEGO MINDSTORMS, *in* 'Proceedings of the 2000 IEEE International Conference on Robotics and Automation', IEEE, San Francisco, CA, pp. 813–819.

Mahadevan, S. & Theocharous, G. (1998), Optimizing Production Manufacturing using Reinforcement Learning, *in* 'Proceedings of the Eleventh International FLAIRS Conference', AAAI Press, pp. 372–377.

Mann, H. B. & Whitney, D. R. (1947), 'On a Test of Whether One of Two Random Variables is Stochastically Larger than the Other', *Annals of Mathematical Statistics* **18**(1), 50–60.

Matarić, M. J. (1992), Behavior-Based Systems: Key Properties and Implications, *in* 'IEEE International Conference on Robotics and Automation, Workshop on Architectures for Intelligent Control Systems', Nice, France, pp. 46–54.

Matarić, M. J. (1994), Interaction and Intelligent Behavior, PhD thesis, Massachusetts Institute of Technology.

Matarić, M. J. (1997*a*), 'Behavior-Based Control: Examples from Navigation, Learning, and Group Behavior', *Journal of Experimental and Theoretical Artificial Intelligence* **9**(2–3), 323–336. Special issue on Software Architectures for Physical Agents.

Matarić, M. J. (1997*b*), 'Behavior-Based Control: Examples from Navigation, Learning, and Group Behavior', *Journal of Experimental and Theoretical Artificial Intelligence* **9**(2–3), 323–336.

Matarić, M. J. (1999), Behavior-Based Robotics, *in* R. A. Wilson & F. C. Keil, eds, 'The MIT Encyclopedia of Cognitive Sciences', MIT Press, pp. 74–77.

Mathisen, H. C. (1943), 'A Method of Testing the Hypothesis that Two Samples are from the Same Population', *Annals of Mathematical Statistics* **14**(2), 188–194.

McCallum, A. K. (1996), Reinforcement Learning with Selective Perception and Hidden State, PhD thesis, University of Rochester, Department of Computer Science, Rochester, New York.

Michaud, F. & Matarić, M. J. (1998), 'Learning from History for Behavior-Based Mobile Robots in Non-stationary Conditions', *Autonomous Robots* **5**(3–4), 335–354.

Mickey, M. R. (1975), 'Approximate Tail Probabilities for Student's t Distribution', *Biometrika* **62**(1), 216–217.

Mitchell, T. M. (1997), *Machine Learning*, The McGraw-Hill Companies, Inc.

Mood, A. M. (1954), 'On the Asymptotic Efficiency of Certain Nonparametric Two-Sample Tests', *Annals of Mathematical Statistics* **25**(3), 514–522.

Page, E. (1977), 'Approximations to the Cumulative Normal Function and its Inverse for Use on a Pocket Calculator', *Applied Statistics* **26**(1), 75–76.

Parker, L. E. (1992), Adaptive Action Selection for Cooperative Agent Teams, *in* 'From Animals to Animats: International Conference on Simulation of Adaptive Behavior', MIT Press, pp. 442–450.

Parker, L. E. (1994), Heterogeneous Multi-Robot Cooperation, PhD thesis, MIT.

Paulson, E. (1942), 'An Approximate Normalization of the Analysis of Variance Distribution', *Annals of Mathematical Statistics* **13**(2), 233–235.

Peizer, D. B. & Pratt, J. W. (1968), 'A Normal Approximation for Binomial, F, Beta, and Other Common, Related Tail Probabilities, I', *Journal of the American Statistical Association* **63**(324), 1416–1456.

Pirjanian, P. (1998), Multiple Objective Action Selection & Behavior Fusion using Voting, PhD thesis, Institute of Electronic Systems, Alborg University, Denmark.

Pirjanian, P. & Matarić, M. J. (2000), Multi-Robot Target Acquisition using Multiple Objective Behavior Coordination, *in* 'Proceedings of the 2000 IEEE International Conference on Robotics and Automation', IEEE, San Francisco, CA, pp. 2696–2702.

Pirjanian, P., Christensen, H. I. & Fayman, J. A. (1998), 'Application of voting to fusion of purposive modules: An experimental investigation', *Robotics and Automation* **23**, 253–266.

Pratt, J. W. (1968), 'A Normal Approximation for Binomial, F, Beta, and Other Common, Related Tail Probabilities, II', *Journal of the American Statistical Association* **63**(324), 1457–1483.

Prescott, P. (1974), 'Normalizing Transformation of Student's t Distribution', *Biometrika* **61**(1), 177–180.

Press, W. H., Teukolsky, S. A., Vetterling, W. T. & Flannery, B. P. (1992), *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press.

Quenouille, M. H. (1953), *The Design and Analysis of Experiment*, Griffin, London.

Rabiner, L. R. (1989), 'A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition', *Proceedings of the IEEE* **77**(2), 257–285.

Roberts, F. S. (1976), *Discrete Mathematical Models: With Applications to Social, Biological, and Environmental Problems*, Prentice-Hall, Inc.

Rosenblatt, J., Willams, S. & Durrant-Whyte, H. (2000), Behavior-Based Control for Autonomous Underwater Exploration, *in* 'Proceedings of the 2000 IEEE International Conference on Robotics and Automation', IEEE, San Francisco, CA, pp. 920–925.

Ross, S. M. (1992), *Applied Probability Models with Optimization Applications*, Dover Publications, Inc., New York.

Schmeiser, B. W. (1979), 'Approximations to the Inverse Cumulative Normal Function for Use on Hand Calculators', *Applied Statistics* **28**(2), 175–176.

Scott, A. & Smith, T. M. F. (1970), 'A Note on Moran's Approximation to Student's t', *Biometrika* **57**(3), 681–682.

Shannon, C. E. & Weaver, W. (1963), *Mathematical Theory of Communication*, Univerisity of Illinois Press.

Siegel, S. & Castellan, N. J. (1988), *Nonparametric Statistics for the Behavioral Sciences*, second edn, McGraw-Hill.

Smithers, T. (1995), What the Dynamics of Adaptive Behavior and Cognition Might Look Like in Agent-Environment Interaction Systems, *in* 'Practice and Future of Autonomous Agents', Mt. Verita, Switzerland.

Sutton, R. S., Precup, D. & Singh, S. (1999), 'Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning', *Artificial Intelligence* **112**, 181–211.

Tan, K.-H. & Lewis, M. A. (1997), 'Virtual Structures for High-Precision Cooperative Mobile Robot Control', *Autonomous Robots* **4**(4), 387–403.

Vaughan, R. T., Støy, K., Sukhatme, G. S. & Matarić, M. J. (2000), Whistling in the Dark: Cooperative Trail Following in Uncertain Localization Space, *in* 'Proceedings of the Fourth International Conference on Autonomous Agents', ACM Press: New York, pp. 187–194.

Wallace, D. L. (1959), 'Bounds on Normal Approximations to Student's and the Chi-Square Distributions', *Annals of Mathematical Statistics* **30**(4), 1121–1130.

Wang, G. & Mahadevan, S. (1999), Hierarchical Optimization of Policy-Coupled Semi-Markov Decision Processes, *in* 'Proceedings of the Sixteenth International Conference on Machine Learning', San Francisco, CA: Morgan Kaufmann Publishers, Bled, Slovenia, pp. 464–473.

Werger, B. B. & Matarić, M. J. (1996), Robotic "Food" Chains: Externalization of State and Program for Minimal-Agent Foraging, *in* 'From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior', The MIT Press: Cambridge, Massachusetts, pp. 625–634.

Whitehead, S. D. & Ballard, D. H. (1991), 'Learning to Perceive and Act by Trial and Error', *Machine Learning* **7**(1), 45–83.

Whitehead, S. D. & Lin, L.-J. (1995), 'Reinforcement Learning of Non-Markov Decision Processes', *Artificial Intelligence* **73**(1–2), 271–306.

Wilcox, R. R. (1997), *Introduction to Robust Estimation and Hypothesis Testing*, Statistical Modeling and Decision Science, Academic Press.

Wilcoxon, F. (1945), 'Individual Comparisons by Ranking Methods', *Biometrics* **1**, 80–83.

Appendix A

Design and Evaluation of Robust Behavior-Based Controllers

This appendix is designed to complement the main body of the dissertation by demonstrating how robust behavior-based controllers for mobile robots can be designed and evaluated. The preceding chapters of the dissertation assumed the existence of (or ability to implement) a basic behavior-based controller for the foraging task. This basic controller (Chapter 2) served as the foundation for applications using on-line, AMM-based evaluations of the agent-environment interaction dynamics. This appendix provides a more complete understanding of how to construct and quantitatively analyze a behavior-based controller than do the preceding chapters. It also contrasts in the use of off-line evaluation, rather than on-line evaluation. This appendix is designed to be self-contained and can be read independently of the rest of the dissertation.

In this appendix, we demonstrate the effectiveness of behavior-based control in facilitating the development and evaluation of multi-robot controllers that are: (1) robust to robot failures, and (2) easily modified to facilitate development of the controller variation that sufficiently satisfies the design requirements for the task. Our experimental focus here is *distributed multi-robot collection*, a class of tasks that includes de-mining and toxic waste clean-up. (This appendix uses a slightly different terminology, referring to the ethologically-named foraging task of the previous chapters as the collection task.) We demonstrate a basic, *homogeneous* multi-robot controller for the collection task, then show how to easily derive two *heterogeneous*, spatio-temporal variations with markedly different performance properties. We evaluate the desirability of these controllers with respect to design requirements involving inter-robot interference, time-to-completion, and energy expenditure. The data for evaluation come from experiments using four physical mobile robots performing the three variations of the collection task.

A.1 Introduction

Designing and implementing robust controllers for multiple interacting mobile robots is considered something of a black art, often involving a great deal of reprogramming and parameter adjustment. It is difficult enough to develop a multi-robot controller that functions only under the ideal conditions of little noise and no robot failures. The fact that such ideal conditions do not often exist, even in a laboratory setting, places certain practical requirements on the multi-robot controller. In particular, the controller must exhibit group-level robustness to noise and robot failures. This is especially important when physical human intervention is difficult (e.g., a toxic waste spill) or impossible (e.g., an extraterrestrial mission).

Additional design requirements for the controller arise from the fundamental, constrained resources of the system, including energy, time, and the number of robots. Untethered mobile robots are generally powered by batteries and can only perform a limited amount of work before needing recharging. Minimizing energy utilization is thus often required in domains, such as space exploration, where recharging is expensive, difficult, or time consuming. In time-critical domains, such as search and rescue, the requirement is for expedient execution of the task. Additionally, regardless of the domain, the fragility of the robots may require the controller to maintain both robot-object and inter-robot collisions at a minimum.

For a given task environment and set of robots, the requirements for the controller may not be independent but instead arise as tradeoffs. For example, minimizing both time and inter-robot collisions may not be possible since faster moving robots are less likely to properly sense each other and thus, more likely to collide. Different controller variations may have to be tested and compared in order to select one that sufficiently satisfies the requirements, given the tradeoffs among them. This places an additional requirement on the controller, namely that it be easily modifiable. The testing and comparison of the variations could potentially be accomplished analytically if an adequate model of the system were developed (a significant challenge in itself), or in simulation (potentially less difficult). In either case, the desire to be able to easily modify the controller remains. Our assumption in this work is that neither an adequate (i.e., very high fidelity) model nor simulation of the physical multi-robot collection task need exist, and thus, we performed all tests directly on physical robots.

The controllers we present in this appendix are designed to address the requirments above. Specifically, they exhibit group-level robustness to robot failures and noise, and are easily modified. Our focus is on the domain of distributed multi-robot collection (foraging) tasks, including toxic waste clean-up and de-mining. We present a basic *homogeneous* controller for the collection task in which all of the robots have identical behavioral repetoirs and work concurrently. We then derive two *heterogeneous* variations, *pack* and *caste*, which respectively modify the robots' temporal and spatial interactions. Finally, we evaluate and compare the performance of the controllers using three spatio-temporal criteria: inter-robot collisions, distance traveled by each robot, and time-to-completion for the task. The latter two criteria also provide an indication of the energy expenditure of the robots. The data for evaluation come from experiments we conducted using four physical mobile robots performing the three variations of the collection task.

Section 2 describes the structure of the collection task and the group of physical mobile robots that performed it. Section 3 then presents the details of the homogeneous controller including the behaviors it contains and how it achieves robustness. Section 4 considers spatio-temporal interactions between robots, especially physical interference, and motivates the two interference-modifying heterogeneous controller versions, pack and caste, presented in Sections 5 and 6. Section 7 presents an analysis of the controllers using data from physical experiments, and provides a comparative evaluation. Finally, a summary is presented in Section 8.

A.2 The Collection Task

The controllers we present implement versions of a multi-robot collection (foraging) task, a prototype for various applications including distributed solutions to de-mining, toxic waste clean-up, and terrain mapping. We present the general structure of the collection task, our multi-robot test-bed, and then the controllers.

A.2.1 Task Structure

We define the collection task as a two-step repetitive process in which:

1. n ($n \geq 1$) robots search designated regions of space for certain objects, and
2. once found, these objects are brought to a goal region using some form of navigation.

A region in the task is any contiguous, bounded space (in the case of mobile robots, a planar surface) which the robots are capable of moving across. There are three mutually-exclusive, non-overlapping types of regions:

- search regions, S, containing a number, p, of objects, a fraction of which must be delivered to a goal region;
- goal regions, G, where objects are delivered;
- and, optionally, empty regions, E, that contain no objects and are not goal regions.

The only restrictions placed on the configuration of regions for the collection task are: that there be at least one search and one goal region, and that the union of all the regions be contiguous. Figure A.1 gives two examples of possible valid region configurations for the collection task.

The specific configuration we used is shown in Figure A.2. The experiments were performed in an 11 $\times$ 14 foot rectangular enclosure (the Corrall). The search region, S, is approximately 126 square feet and has $p = 27$ small metal cylinders (pucks) evenly distributed throughout. The goal region G, also called *Home*, is a ninety degree sector of a circle with a radius of 2 feet, located in one corner of the Corrall. Finally, there is a 25 square foot empty region, E, separating the

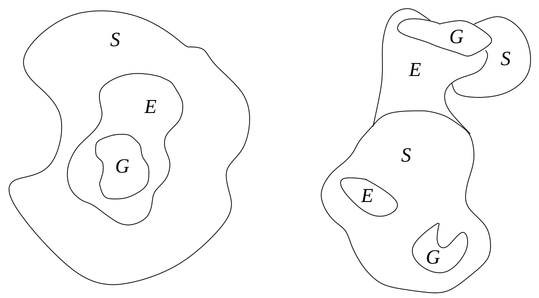


Figure A.1: Two example region configurations for the collection task.

search and goal regions. E is composed of the Boundary and Buffer zones, whose functions will be described in the next section. $n = 4$ robots are used in the experiments.

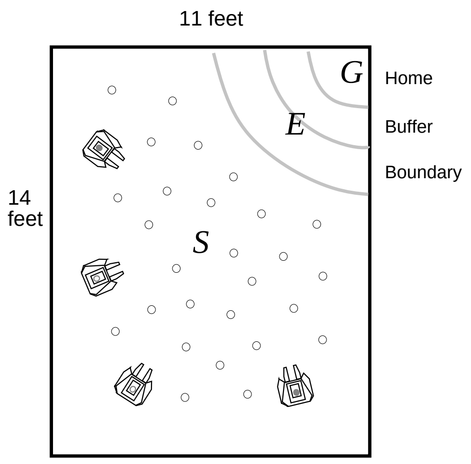


Figure A.2: Actual configuration used in the collection task.

A.2.2 The Robots

Four IS Robotics R2e robots were used (Figure A.3). Each is a differentially-steered base equipped with two drive motors and a two-fingered gripper. The sensing capabilities of each robot include piezo-electric contact (bump) sensors around the base and in the gripper, five infrared (IR) sensors around the chasis and one on each finger for proximity detection, a color sensor in the gripper, a radio transmitter/receiver for communication and data gathering, and an ultrasound/radio triangulation system for positioning (Figure A.4). The robots are programmed in the Behavior Language (Brooks 1990), a parallel, asynchronous, behavior-based programming language inspired by the Subsumption Architecture (Brooks 1986). The main computational power on each robot is a single

Motorola 68332 16-bit microcontroller running at 16 MHz. Even though computationally impoverished by today's standards, the processing capabilities have proven to be adequate for most tasks we have envisioned, helping to show that robust, effective control need not be computationally expensive. Perhaps the greatest drawback of the 68332 is its lack of floating point computation, which, for example, influences our calculation of heading, described in the following section.

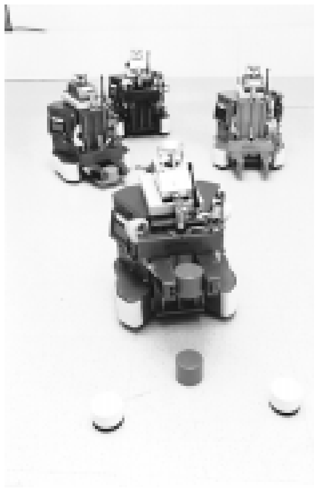


Figure A.3: The four R2e robots used in the experiments.

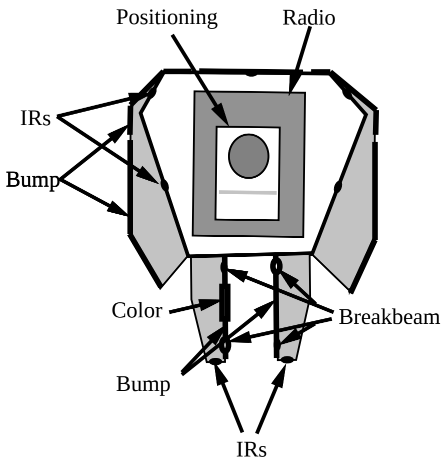


Figure A.4: The sensor configuration of an R2e robot.

A.2.3 Behavior-Based Control

The work presented in this appendix is couched in the framework of distributed behavior-based control (Brooks 1991, Matarić 1992). Behavior-based control has proven to be an effective paradigm for developing single-robot and multi-robot controllers (Arkin 1998). In behavior-based control, the robot controller is organized as a collection of event-driven modules, called behaviors, that receive inputs from sensors and/or other behaviors, process the input, and send outputs to actuators and/or other behaviors. Each behavior generally serves some independent function, such as *avoiding* obstacles or *homing* to a goal location. All behaviors in a controller are executed in parallel, simultaneously receiving inputs and producing outputs. An action selection mechanism prevents conflicts when multiple outputs are sent to actuators or other behaviors (Pirjanian 1998). The controllers presented in this appendix demonstrate the suitability of the behavior-based paradigm for designing robust and modifiable multi-robot controllers.

In the next section, we present our initial, *homogeneous* controller for the collection task, followed later by two heterogeneous variations, *pack* and *caste*.

A.3 The Homogeneous Controller

In this section, we present the first of three behavior-based controllers we implemented. This first controller performs a homogeneous version of the collection task where the robots' behavioral repetoirs are identical, and the robots act concurrently and independently.

The overall structure of the controller is presented in Figure A.5. In the figure, the rounded rectangles represent the robot's sensors, with sensor values being transmitted to behaviors along the dotted lines. The behaviors themselves are drawn as ellipses with text in one of three font styles: italics for behaviors that only receive sensor inputs; bold for behaviors that send actuator outputs; and bold-italics for behaviors that do both. The dashed lines represent commands sent by behaviors to the actuators (rectangles), and the solid lines represent control signals sent between behaviors. These control signals include: inhibition signals that temporarily disable behaviors, or do so permanently until the inhibition is lifted; information about the state of the behaviors; and signals indicating that a behavior should perform a certain action. These control signals establish the hierarchy of actuator commands shown at the right of the diagram. The $\otimes$ represents behavior selection and indicates that only one of relevant actuator command pathways is active at any time. The $\odot$ represents a Subsumption-style priority scheme with the actuator command coming from above taking precedence (Brooks 1986). The hierarchy of command pathways in the diagram illustrates that behavior arbitration is the action selection mechanism for the controller. The next section presents, in detail, the function of the each behavior in the controller, and the structure of the inter-behavior command pathways. The subsequent section discusses the group-level robustness achieved by this controller.

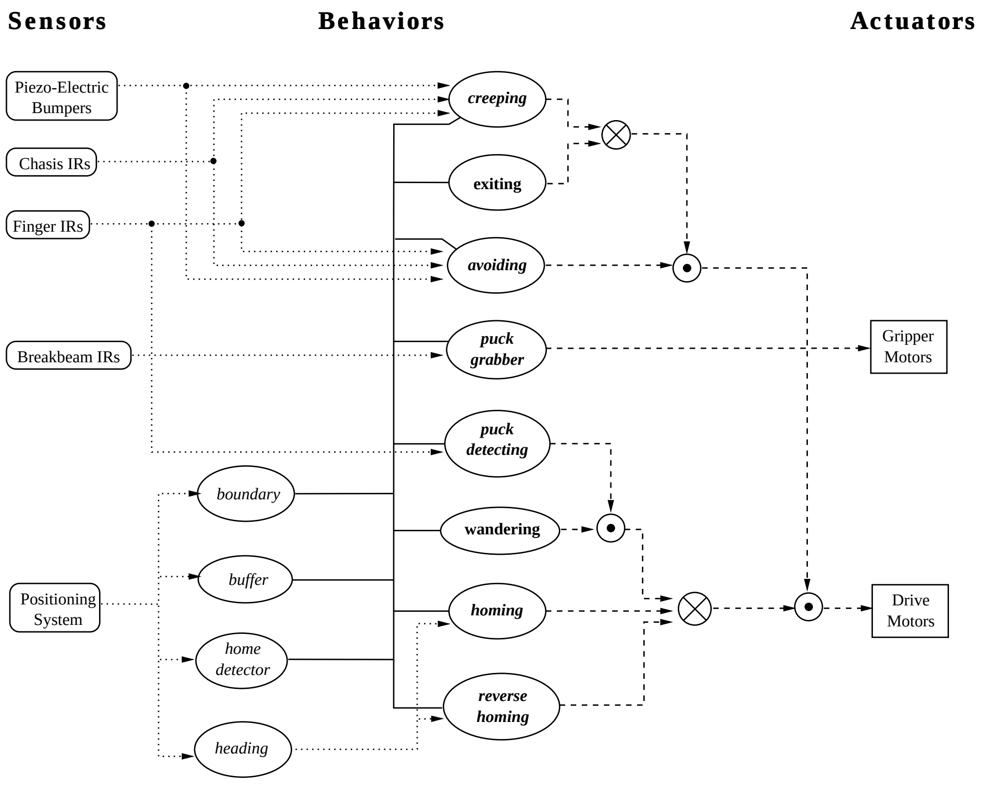


Figure A.5: The homogeneous controller for the collection task. Rounded rectangles represent the robot's sensors, ellipses represent behaviors, and rectangles represent actuators. Sensor values are transmitted along dotted lines, actuator commands along dashed lines, and inter-behavior control signals along solid lines. The symbol $\otimes$ represents behavior selection and $\odot$ represents Subsumption-style precedence.

A.3.1 Behaviors

In order to provide a clear picture of the interaction between behaviors, we describe the individual behaviors of the controller in an order that mirrors the progression of the task as the robot performs it. The following twelve behaviors constitute the collection task:

1) *avoiding*: This behavior avoids any object (including other robots) detected by the IR sensors and deemed to be in the path of, or about to collide with, the robot. If the robot has already collided with an object, as detected by the contact sensors, it steers away from it. This behavior is critical to the safety of the robot and therefore takes precedence over most of the behaviors that control the drive motors (*puck detecting*, *wandering*, *homing*, *reverse homing*).

2) *wandering*: The robot moves forward and, at random intervals, turns left or right through some random arc. Using this behavior, the robot searches the region for pucks.

3) *puck detecting*: If an object is detected by the front IR sensors while *wandering*, this behavior, by lifting the gripper, determines whether the object is short enough to be a puck, or whether it

is an obstacle that must be avoided. If it is a puck, the robot carefully approaches the object and attempts to place it between its fingers. Otherwise, the robot performs *avoiding*.

4) *puck grabber*: When a puck enters the fingers and is detected by the breakbeam IR sensors, this behavior grasps it and raises the fingers. Raising the fingers above puck height prevents the robot from unnecessarily avoiding pucks while *homing*, and allows the robot to collect up to about four additional pucks with its base.

5) *homing*: If carrying a puck, the robot moves towards the designated goal location, Home. While *homing*, *avoiding* can take precedence in order to avoid obstacles.

6) *boundary*: This behavior monitors how the robot enters the Boundary region. If the robot enters this region without a puck, it returns it to the search region using *reverse homing*. If carrying a puck, the robot is allowed to enter this region and proceed towards Home (see Figure A.2). This behavior prevents the robot from collecting pucks that have already been delivered.

7) *buffer*: This behavior monitors entry into the Buffer region. Entering this region triggers the activation of the *creeping* behavior.

8) *creeping*: A refined combination of the *homing* and *avoiding* behaviors designed to carefully bring the robot to the very corner of the Corrall where Home is located and where the pucks must be delivered. Under *creeping*, the robot moves more slowly and uses its IR sensors at a closer range appropriate for working within the corner. The standard versions of *homing* and *avoiding* would conflict in a confined corner situation, since *avoiding* would perceive the goal corner as an obstacle and attempt to move the robot away from it. *Creeping* takes precedence over *avoiding* since it already incorporates a version of this behavior.

9) *home detector*: A monitoring behavior for entry into the Home region. Upon entering this region, *home detector* sends a signal to *puck grabber* to release the puck.

10) *exiting*: Entering the Home region triggers this behavior which moves the robot several inches backwards, then performs a 180-degree turn in place. This behavior also sends the signal that lowers the gripper. When *exiting* terminates, the robot remains within the Boundary region without a puck. This in turn triggers the *boundary* behavior to begin *reverse homing*.

11) *reverse homing*: Starting from within the Boundary region, this behavior performs the opposite of *homing*, moving the robot out into the search region. This behavior is essentially identical to *homing* except that the goal location is set to the corner of the Corrall opposite Home. Once the Boundary region has been left, *reverse homing* becomes inactive and the robot once again begins searching for pucks using *wandering*.

12) *heading*: This behavior processes the positioning system data and provides approximate heading values for the *homing* and *reverse homing* behaviors. The positioning system supplies the robot's current (x, y) position at approximately 1–2 Hz. Consecutive position values, (x_0, y_0) and (x_1, y_1), are used in an approximate integer-based calculation of $\arctan(\frac{y_1-y_0}{x_1-x_0})$ adjusted for the quadrant of the angle to provide one of sixteen possible sector headings. The accuracy of this heading calculation is usually within one sector of the true heading, but may be far worse when the robot turns in place. Frequent updates of the heading, with little reliance by the other behaviors on any one heading value, help to compensate for the inaccuracies. (An alternative is to use a physical

compass for heading data. In our lab, however, the high variance in magnetic fields makes this inviable.)

A.3.2 Robustness

In the above described homogeneous controller, group-level robustness is a direct result of the robots behaving identically and independently. No noise-susceptible, or time-critical, radio communication that could be a source of fragility in the system is necessary. Each robot must individually manage the noise and uncertainty associated with its sensors and actuators, and the complexity of a dynamic and basically unknown environment. (Our controller, as is true for most behavior-based controllers, accommodates noise and uncertainty by tightly coupling sensing to action so that no great reliance is placed on any one sensor reading.) The partial, or even complete, failure of any one robot, or a subset of them, does not debilitate the entire group. As long as there is one functioning robot, the task can be accomplished.

As discussed previously, in addition to exhibiting group-level robustness, a multi-robot controller should be easy to modify in order to facilitate the search for an acceptable variation. The desirability of the controller must be measured with respect to any design requirements, such as time-to-completion of the task, energy consumption, or the amount of interference exhibited. Thus, before we present the variations of our homogeneous controller, we discuss the key diagnostic parameters used in evaluation. Our focus here is on inter-robot interference, specifically physical collisions between robots. The goal that motivates the modification of the homogeneous controller is minimization of such interference. The next section provides a discussion of interference and the two spatio-temporal solutions to it which provide the basis for our heterogeneous controller variations.

A.4 Spatio-Temporal Interactions

In this section, we discuss the nature of physical inter-robot interference (i.e., collisions), and how a multi-robot system may be modified to manipulate this interference. Our discussion here provides the motivation for the two controller variations, *pack* and *caste*, presented later.

Multi-robot systems are by definition physically embodied and embedded in dynamic environments. The types of interference they contain can be distinguished about a physical/non-physical dichotomy. Physical interference manifests itself most overwhelmingly in competition for space. Non-physical interference ranges from the sensory (shared radio bandwidth, crossed infrared or ultrasound sensors) to the algorithmic (the goals of one robot undoing the work of another, competing goals, etc.). Here we focus on physical interference and demonstrate that it is an effective tool for system evaluation and design.

We define the *characteristic interference* of a system at a particular point in space to be the sum, over some finite time period, of all measured interference occurring at that location (see Figure A.6). The result is a surface that can be used to adjust the controller in order to reduce interference and

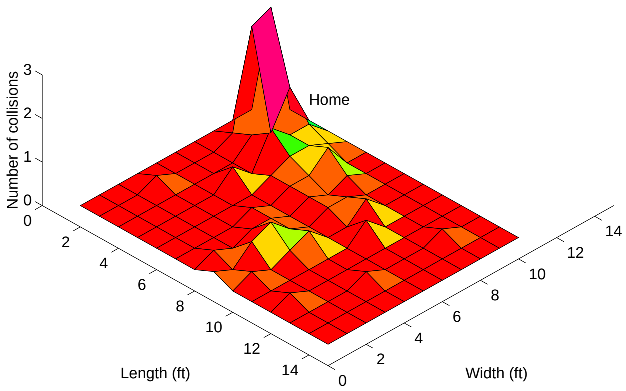


Figure A.6: This plot shows the characteristic interference pattern for the homogeneous implementation of the collection task on four physical robots. The shading, corresponding to the height of the peaks, is clearer when the data support a very fine mesh.

thus modify the system's overall performance. Robot density is a critical factor in characteristic interference. Single-robot systems and systems with density so high as to prevent movement produce no characteristic interference. Systems of interest lie in between the two extremes.

A principled multi-step process of controller modification can be implemented by using characteristic interference as a guide indicating where in the robots' physical interaction, and when, within the lifetime of the task, behaviors should be switched and the task should be divided to modify overall task interference. Multi-robot interactions we focus on are spatio-temporal in nature and fall into four basic categories. Robots may either be in the *same place* (SP) or in *different places* (DP), both of which can occur at the *same time* (ST) or at *different times* (DT), resulting in four forms of interaction: SPST, SPDT, DPST, and DPDT.

Physical interference fits into the SPST category, covering the case when two or more robots try to occupy the same location at the same time. The other three categories are useful for deriving and fine-tuning controllers that modify SPST interactions. For two of these categories, we implemented and tested a corresponding controller. The SPDT category is associated with our *pack* controller, a temporal modification to the homogeneous controller, while DPST is associated with our *caste* controller, a spatial modification of the homogeneous controller scheme. The DPDT category represents the case where there is little possibility of physical interaction. For example, the robots may occupy non-contiguous regions of space, or only one robot at a time may be activated. Since our focus is on controllers for multiple robots interacting to accomplish a task, the DPDT category does not provide an acceptable solution for interference management.

Figure A.6 presents the characteristic interference pattern for the homogeneous implementation showing the number of collisions between robots within the Corrall. The data for the plot are an

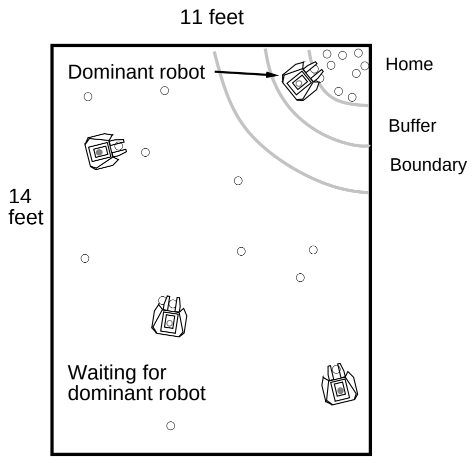


Figure A.7: The pack variation of the collection task.

average of the collisions observed over five trials with the completion criterion defined as collecting 14 of the 27 pucks at Home. The figure shows high levels of interference near Home resulting from multiple robots simultaneously attempting to deliver pucks. We thus seek to modify the controller in order to reduce this interference using our two spatio-temporal variations, pack and caste. We present a more detailed comparative evaluation of interference later in the Analysis section.

A.5 The Pack Controller

In the pack controller, as in the homogeneous version, all individuals have identical behaviors and activation conditions. Unlike the homogeneous controller, however, the robots do not act concurrently and independently. Instead, a dominance hierarchy is imposed, based on some functional criterion such as the robots' different capabilities, or on an arbitrary assignment scheme such as robot ID, if the robots are functionally identical (as are ours).

The dominance hierarchy induces a temporal structure on the task by allowing only one of the robots to deliver a puck at any time. All of the robots may search for pucks in parallel, as in the homogeneous implementation, but if two or more robots simultaneously find pucks, the one highest in the hierarchy is allowed to deposit its pucks first. The other robot(s) cannot proceed until the first has finished delivering its pucks and has left the Boundary region (Figure A.7). This scheme introduces temporal heterogeneity to the homogeneous version, and thus corresponds to SPDT (or temporal) arbitration of SPST interactions.

The pack strategy requires that some form of dominance hierarchy be assigned and that dominance rank be recognized between the robots. In our case, rank was communicated over the radios, but in other implementations it could be based on physical characteristics that can be sensed directly.

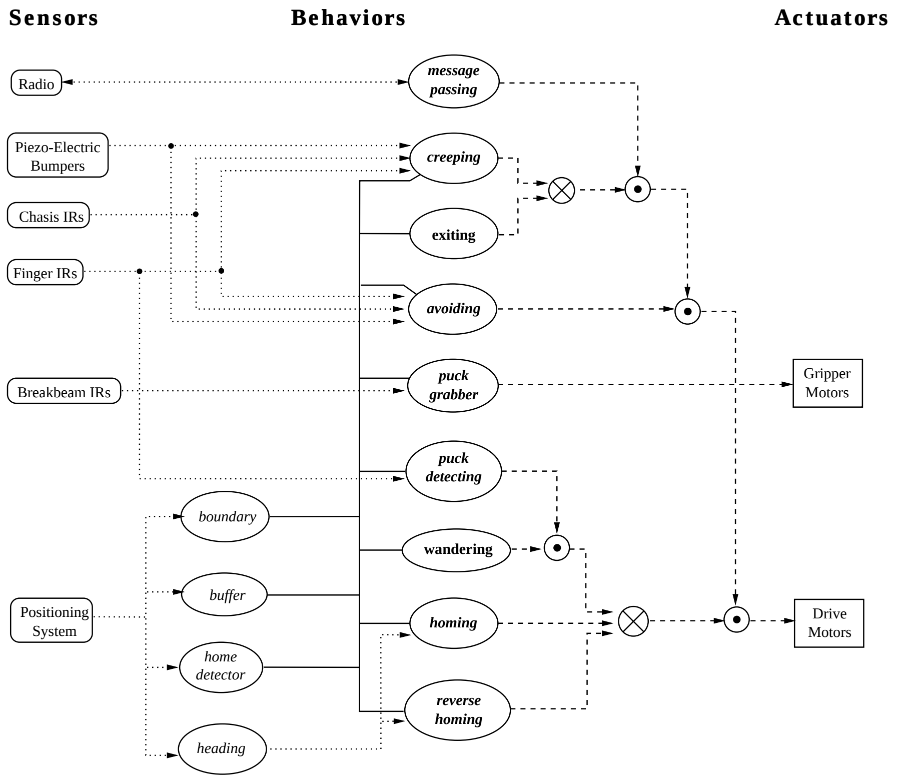


Figure A.8: The pack version of the controller for the collection task.

A.5.1 The "message passing" Behavior

Figure A.8 presents the controller for the pack implementation. This controller is almost identical to the homogeneous controller (Figure A.5), except that it includes a high-precedence *message passing* behavior. The function of *message passing* is to send the robot's status, specifically whether it is delivering a puck, to the other robots, and in turn monitor the status of the other robots. When a robot finds a puck, *message passing* places the robot into a wait state with the motors off and enters the following communications routine:

1. Wait two communication cycles (approximately 6 seconds) to accumulate the most current status information from each robot.

2. If after (1) above, no other robot is currently delivering a puck, transmit the desire to do so. Otherwise return to (1).

3. Wait three communication cycles (approximately 10 seconds) for synchronization with the other robots.

4. If after (3) above, no other robot wishes to deliver a puck, or any that do are less dominant, then proceed to deliver the puck and inform the other robots when finished. Otherwise, return to (1).

We now consider how *message passing* ensures robustness, one of the requirements for our multi-robot controller.

A.5.2 Robustness

As we have discussed, it is important that multi-robot controllers be robust to noise and robot failures. Similar to the homogeneous controller, the pack controller accommodates robot failures by having each robot able to accomplish the entire task. Unlike the homogeneous controller, the coordinated hierarchy of the pack controller requires special measures by the *message passing* behavior to ensure robustness. If a robot fails while searching for a puck, no special measures are required since no other robot is waiting upon its actions. If, however, the robot fails while delivering a puck, the other robots must be informed so as not to wait indefinitely. The failed robot can send such a message if it is able to detect the failure (a difficult problem in itself). Otherwise, some external agent, such as a human operator, can send the message.

We use a somewhat different approach in our experiments. Whenever a robot fails, it is shut down and restarted by a human operator. (In hazardous conditions, it may be possible to re-pair/restart the robots remotely.) During this restart period, the waiting robots receive no communications from the failed delivering robot. The robots consider such periods of protracted radio silence as an indication of the robot's failure, and once again enter into the communications routine above. Once the failed robot has restarted and begins communicating, it is seamlessly incorporated back into the hierarchy. Since the communications routine only uses relative dominance to decide which robot should deliver a puck, it easily accommodates the attrition or addition of robots.

Another advantage of our communications routine above is that the use of *radio silence failure detection* helps provide group-level robustness to radio noise. As noise levels increase, communication between the robots becomes increasingly difficult. This may lead to protracted periods of radio silence that are incorrectly interpreted as robot failures. In such a situation, two or more robots may deliver pucks at the same time. The degradation of the hierarchy, however, is what prevents the failure of the entire group. Even if the radio system were to fail completely, the task would still be accomplished because every robot would consider every other robot as having failed. Thus, the pack controller would degenerate into the homogeneous controller. We posit that such graceful degradation in group structure, without jeopardizing the entire task, is an important property of controllers for unknown and dynamic environments.

A.5.3 Interference

Figure A.9 shows the characteristic interference pattern for the pack controller, averaged over 5 trials. The completion criterion was identical to the homogeneous case: delivering 14 of the 27

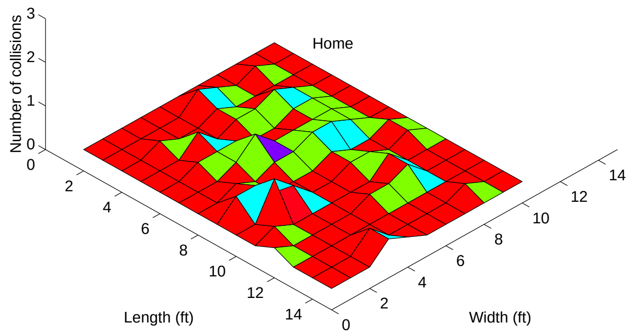


Figure A.9: This plot shows the characteristic interference pattern for the pack implementation of the collection task on four physical robots. The shading corresponds to the height of the peaks.

pucks to Home. As is clear from a comparison to the characteristic interference of the homogeneous controller (Figure A.6), the pack controller has reduced interference near Home, as desired.

Not only is the pack controller successful in reducing interference, it is also attractive in its ease of implementation. The pack variation is simply the homogeneous controller with the addition of the dominance hierarchy induced by the *message passing* behavior. Such ease of implementation supports our requirement that controllers be easy to modify.

A.6 The Caste Controller

In a caste controller, the group of robots differentiates into two or more sub-groups (castes), each of which acts concurrently and independently, but occupies different regions of the task space. The goal is to manipulate interference by an appropriate division of the task space, and assignment of the castes to the sub-regions. This spatial separation of castes limits physical interactions to the territorial boundaries. The caste scheme introduces spatial heterogeneity and thus corresponds to DPST arbitration of SPST interference.

Unlike the homogeneous and pack strategies, the sub-groups of robots in the caste strategy may have different behavioral repetoirs. Thus, in addition to spatial heterogeneity, a caste controller may also exhibit behavioral heterogeneity. Indeed, that is the case with the caste implementation we present in this section. It consists of two sub-groups: the Search Caste, comprised of three robots which find pucks and bring them near Home, and the Goal Caste, comprised of one robot which brings the pucks the rest of the way to Home (Figure A.10). Each of the two castes has a different controller.

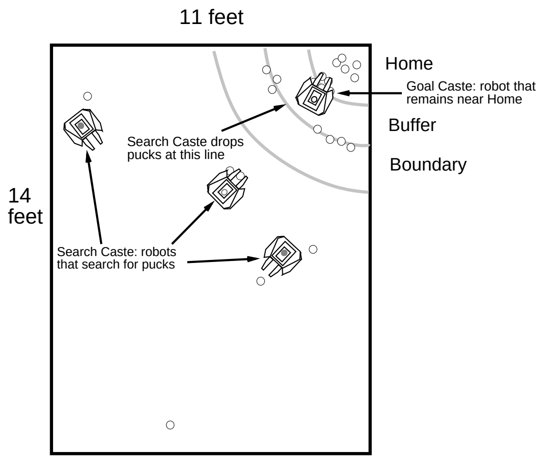

11 feet
14 feet
Home
Goal Caste: robot that remains near Home
Buffer
Boundary
Search Caste drops pucks at this line
Search Caste: robots that search for pucks

Figure A.10: The caste variation of the collection task.

A.6.1 The Search Caste

In our implementation, three of the four R2e robots, comprising the Search Caste, have behavior sets similar to the homogeneous implementation. Each robot searches the region S for pucks, but delivers them to the line separating the Boundary and Buffer zones, rather than all the way to Home. Figure A.11 presents the controller for the Search Caste. It is identical to the homogeneous controller (Figure A.5), except that it lacks the *creeping* behavior. This more refined combination of *homing* and *avoiding*, designed to bring the robots to the corner of the Corrall, is no longer necessary since pucks are not brought to the corner. The *buffer* behavior is also removed from the controller because it is not needed to activate *creeping*.

A.6.2 The Goal Caste

The Goal Caste consists of one robot that remains in the Home and Buffer regions with the task of transporting to Home the pucks dropped by the Search Caste at the Boundary/Buffer line. The controller for the Goal Caste is presented in Figure A.13. The *sweeping* behavior moves the robot away from Home and performs an arc at the Boundary/Buffer line to "scoop up" any pucks left there (Figure A.12). The *creeping* behavior then carefully moves the robot to Home, where it performs *exiting* to back up and deliver the pucks. The robot then turns in place 180 degrees to once again begin *sweeping*. During the execution of the controller, the gripper remains lifted allowing the concave front region of the robot's base to scoop up multiple pucks.

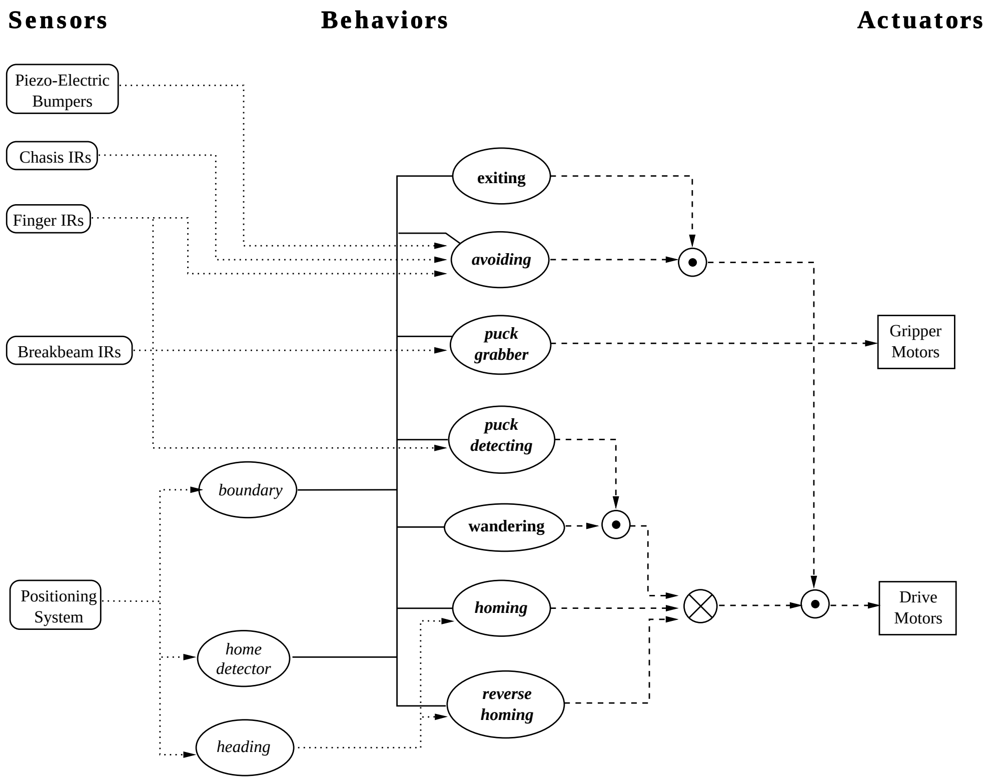


Figure A.11: The controller for the Search Caste, the three-robot subgroup that searches for pucks.

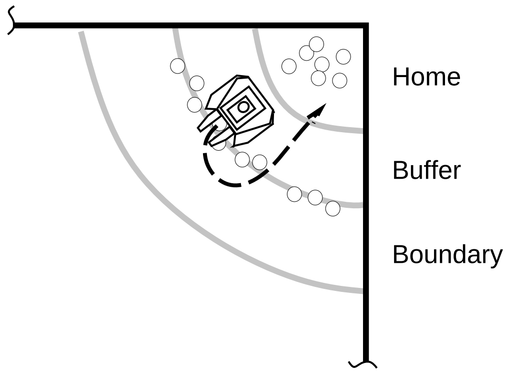


Figure A.12: The *sweeping* behavior of the controller for the Goal Caste.

A.6.3 Robustness and Interference

The controller for the Search Caste shares many of the characteristics of the homogeneous controller. It achieves group-level robustness by maintaining a behaviorally identical group with no reliance on fragile explicit communication. Thus, neither high levels of noise nor the failure of a robot debilitates the entire caste. The Search Caste controller also provides a good example of the ease with which the homogeneous controller can be modified.

One of the keys to robustness in the caste controller is the asynchronicity of interaction between the two castes. The Search Caste must deliver pucks to the Boundary/Buffer line, but the Goal Caste is not dependent upon them arriving at a particular time or in a particular order that may be difficult to ensure in such a complex, stochastic system.

Though not implemented in our caste controller, additional robustness could be added by using a variation of the pack communication protocol to transmit the number of active members of each caste. If one caste were to lose too many individuals, members of the other castes could replace them. For example, if the one robot of the Goal Caste were to fail, a member of the Search Caste could substitute. This scheme, while improving robustness, would require each robot to possess all of the individual caste controllers and be able to switch between them as necessary. Such caste switching would be most robust if duplication of the exact state of the failed robot were not necessary, as would be the case with our controller.

Figure A.14 shows the average characteristic interference over five trials for the caste implementation. The completion criterion was the same as for the homogeneous and pack controllers: 14 of the 27 pucks collected. It is clear from a comparison to the characteristic interference of the homogeneous controller (Figure A.6) that interference near Home is reduced, as was desired. The overall level of physical interference throughout the Corrall, however, is higher with the caste controller.

The following section provides a more detailed quantitative evaluation and comparison of the controllers in terms of interference, as well as time-to-completion and the distance traveled by each robot.

A.7 Analysis

In order to better evaluate the desirability of and tradeoffs between the three controllers — one homogeneous and two heterogeneous — we performed five experimental trials for each, gathering both spatial and temporal data. Initial conditions for all trials were as nearly identical as possible in order to minimize free variables, and the completion criterion was 14 out of 27 pucks collected. For each trial, we gathered data on the time-to-completion of the task, and the location and number of collisions between robots, showing the characteristic interference. We calculated the average total number of collisions for each experiment, providing a relative comparison of the different schemes. Using the positioning system, we also recorded each robot's location at approximately 0.3 Hz in order to examine the distance traveled and path taken by each. Finally, we monitored the activity

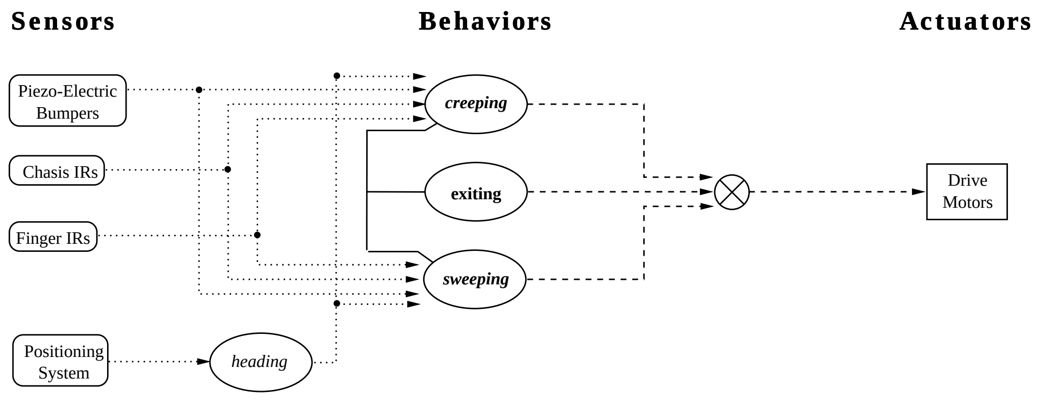


Figure A.13: The controller for the Goal Caste, the one-robot subgroup that brings pucks from the Boundary/Buffer line to Home.

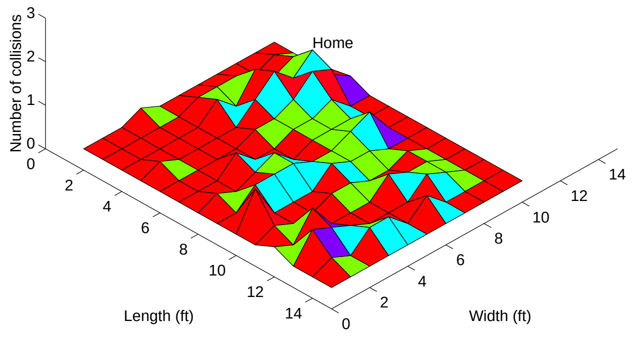


Figure A.14: This plot shows the characteristic interference pattern for the caste implementation of the collection task on four physical robots. The shading corresponds to the height of the peaks.

Controller	Time (sec)	Avoiding (sec)
Homogeneous	549	143
Caste	1447	442
Pack	1081	229

Table A.1: Average time of task completion and average time spent in the *avoiding* behavior for each controller.

of the internal behaviors of the robots. The *avoiding* behavior was of particular interest since it is the one directly invoked by physical interference. We hypothesized that the time spent avoiding would be correlated with the total amount of interference in each of the implementations, and would thus serve as an alternate measure of interference. As shown below, this hypothesis was validated (see Table A.2).

All of the data presented in this section have been analyzed with one or more statistical tests. We have performed hypothesis tests using Student's t, 1-factor analysis of variance (ANOVA), and 2-factor ANOVA, in order to verify that the differences between the results of the implementations were in fact statistically significant. In all cases, these differences were significant with p-values < 0.05.

Our discussion in this section is based on the assumption that the task environment is fixed. Another effective method for altering the spatio-temporal properties considered below is modification of the environment, if such is possible. We could, for example, move Home to the middle of the workspace, thus manipulating properties like interference and time-to-completion.

The majority of this section eschews a quantitative evaluation of heterogeneity, focusing instead on the performance data mentioned above. This is justified in the concluding paragraphs where we discuss the poorly understood relationship between heterogeneity and performance in multi-robot groups.

A.7.1 Interference, Avoiding, and Time

One factor that impacts the total amount of interference observed for each implementation is the time-to-completion of the collection task. One would expect that for any given implementation, the longer the trial continues, the more interference or collisions there would be. One would also expect the total amount of time spent in the *avoiding* behavior to be positively correlated with the time-to-completion. In Table A.1 we see that this is indeed the case. The homogeneous implementation has the shortest time-to-completion and the least amount of time spent avoiding; the pack implementation has the next larger times; and the caste implementation has the largest times over all.

In their current form, the values for time-to-completion and time-spent-avoiding do not provide much useful information about the amount of interference in each controller. We can observe, however, that the amount of time spent in the *avoiding* behavior is composed of the time spent

Controller	Interference (collisions)	Avoiding/Time
Homogeneous	16.4	0.27
Caste	20	0.32
Pack	12.6	0.22

Table A.2: Average amount of interference and average fraction of time spent in the *avoiding* behavior.

Controller	Interference/Time (collisions/sec)
Homogeneous	0.030
Caste	0.014
Pack	0.012

Table A.3: Average amount of interference per unit time for each controller.

avoiding other robots (before, during, and after collisions) and the time spent avoiding everything else. Since the environment (discounting the robots) is identical in every trial, we can assume that the amount of avoidance per unit time attributable to non-robot objects is constant between the implementations. This assumption suggests that any differences in the amount of avoidance per unit time between the implementations would be primarily due to the avoidance of the other robots, possibly during collisions. Thus, we would expect to see a correlation between the average amount of interference observed in each implementation and the ratio of time spent avoiding to total time. In Table A.2 we observe that such a correlation does exist and it is quite large at $\rho = 0.995$. This indicates an important link between avoiding and total time, and suggests that their ratio is a good estimate of relative average interference levels.

Another potentially useful statistic is the amount of interference per unit time. As shown in Table A.3, the pack implementation has the most desirable ratio while the homogeneous implementation has the least.

A.7.2 Distance Traveled

As mentioned previously, the energy expended by the robots in completing the task may be a concern if recharging is time-consuming or difficult. Time-to-completion provides one approximation of energy expenditure, but it can be inaccurate, especially with a controller such as our pack version where robots can be idle for long periods of time. A better approximation is the amount of work accomplished by the robots during the task. Work (W), force (F), and displacement (d) are related through the elementary physics equation

$$W = F \cdot d \cdot \cos\theta$$

Controller	Robot0	Robot1	Robot2	Robot3	Total (ft)
Homogeneous	123	120	113	119	475
Caste	353	370	385	119	1227
Pack	112	145	188	178	623

Table A.4: Average distance (in feet) traveled by the robots for each controller.

or

$$\frac{W}{F \cdot \cos\theta} = d,$$

where θ is the angle between the force and displacement vectors. Since the robots are mechanically identical, we can consider $F \cdot \cos\theta$ to be constant among them. This allows us to compare the work done by the robots solely in terms of d, the distance traveled. Because the robots are identical, d also provides a reasonable, relative indication of the energy expended in performing the work. Finally, it provides a measure of efficiency: the less work required to accomplish the task, the more efficient the controller.

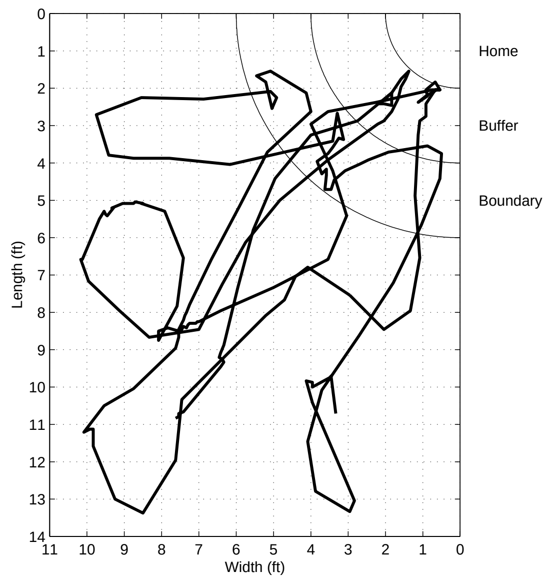
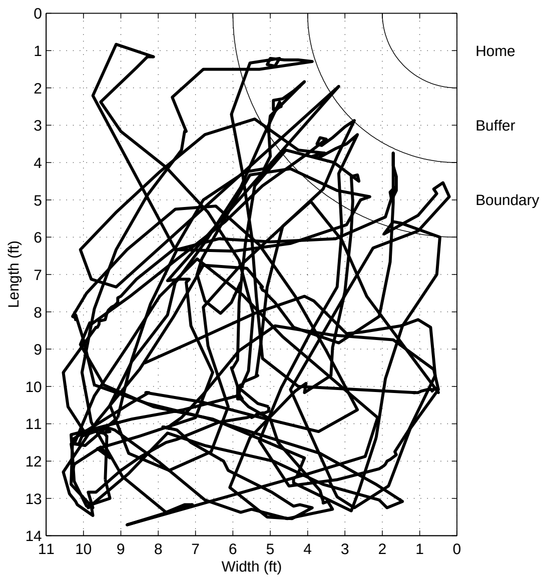


Figure A.15: A typical path taken by one physical robot in the homogeneous controller.

Table A.4 presents the average distance traveled by each robot, and the total over all robots, for each of the three controllers. The values were calculated from the robot position data gathered during the experiments. The results indicate that the homogeneous controller performs the least work in completing the task, and thus is the most efficient, whereas the caste controller performs the most work and is least efficient.



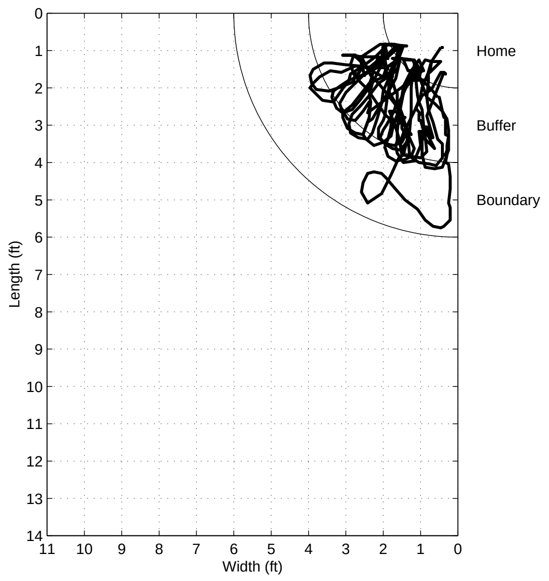
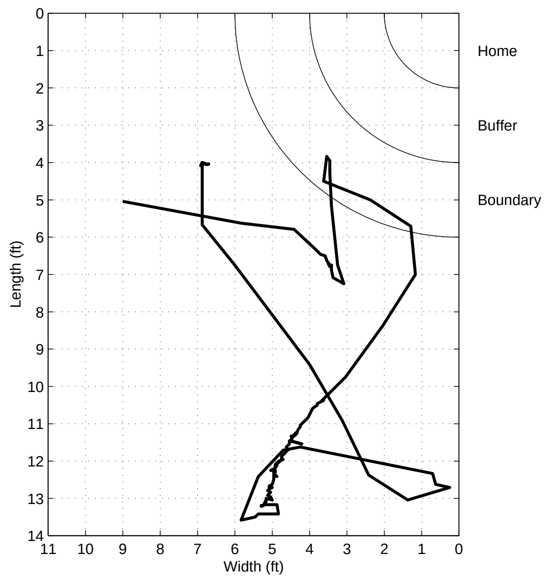


Figure A.16: (Left) A typical path of a physical robot in the Search Caste of the caste controller; (Right) a typical path of the robot in the Goal Caste.

Although the total distances traveled for the three controllers are statistically different, this is not necessarily true of the distances traveled by the individual robots within a controller. This follows intuitively from the structure of the controllers. In the homogeneous controller where all four robots are behaviorally identical, there is no statistical difference in the distances traveled. In the caste controller, Robot0, Robot1, and Robot2, which comprise the Search Caste, travel similar distances, whereas Robot3 of Goal Caste moves significantly less, as might be expected. In the pack controller, one would expect the less dominant robots to travel less since they spend more time waiting for the dominant robots to deliver pucks. Table A.4, with Robot0 as the least dominant and Robot4 as the most dominant, shows that this is the general trend. Although a one-way analysis of variance indicates that there is significant difference among these values, there are too few trials to provide further discrimination using a t-test. (The exception is that Robot0 is shown to travel significantly less than Robot2 and Robot3.)

A more qualitative, visual examination of the execution of the controllers is also possible. Figure A.15 shows a typical path of one robot in the homogeneous controller. It is clear that the robot searches for pucks, delivers several to Home, and sometimes enters the Boundary without pucks and promptly leaves. Figure A.16 (Left) shows a similar path taken by a robot in the Search Caste of the caste controller. The path is much longer than that of the homogeneous controller due to the protracted time of the trial. We also note that the Search Caste very clearly delivers pucks to the Boundary/Buffer line. Figure A.16 (Right) shows the complementary path of the Goal Caste collecting pucks from the Boundary/Buffer line and taking them to Home. Figure A.17 provides a juxtaposition of typical paths taken by the least dominant and most dominant robot of



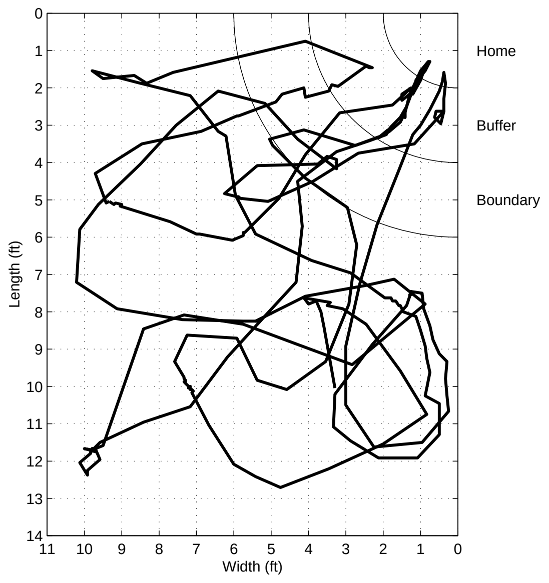


Figure A.17: (Left) A typical path of the least dominant robot of the pack controller; (Right) a typical path of the most dominant robot.

the pack controller. As expected, the most dominant robot has a path (Right) very similar to that of the homogeneous controller. The least dominant robot, however, has a severely stunted path demonstrating the significance of the time it waits for the more dominant robots to deliver their pucks.

A.7.3 Robustness

During the experimental trials for each controller, we had the opportunity to evaluate group-level robustness. The R2e robots used in the experiments are quite fragile and prone to failure from something as simple as a buildup of static electricity corrupting memory or causing the robot's computer to crash. There was seldom a trial without multiple failures requiring the failed robots to be restarted. With the homogeneous controller, we noted very clearly that the failure of one robot did not effect the activity of the others. In the pack controller, the less dominant robots of the hierarchy were able to compensate for the failure of a dominant robot by using the message passing protocol. If a dominant robot failed while delivering a puck (which occurred at least once per trial), the less dominant robots would stop waiting and begin delivering their pucks. In the caste controller, the Search Caste exhibited group-level robustness similar to the homogeneous controller: the failure of one robot did not affect the other members of the caste. In addition, due to the asynchronicity of interaction between the two castes, the failure of the robot in the Goal Caste did not debilitate the activity of the Search Caste.

A.7.4 Evaluation

Using the analyses presented above we can now discuss the relative desirability of the three controllers. All three are desirable in that they exhibit good group-level robustness. The tradeoff between time and interference captures the relative performance. The homogeneous implementation requires the least time but does not result in the least interference, whereas the pack implementation exhibits the least total interference and least interference per unit time, but takes longer overall. Thus, we must decide which criterion is more important or what kind of compromise we wish to make in the final controller choice. If we can sacrifice some performance time for decreased robot interference, then the pack implementation appears to be the best choice. This solution applies to conservative systems where collisions and the possibility of equipment damage outweighs the required time. In contrast, if total time or energy expenditure is the critical factor, such as in domains where the items to be collected are toxic or dangerous, or robot power is limited, then the homogeneous implementation is the better choice. From this analysis we also observe that the caste implementation does not appear to be a satisfactory solution under any criterion, and may be discarded from consideration.

Although our analysis does not identify one controller that is clearly superior in all respects, it does provide information to make an intelligent decision regarding the tradeoffs between the homogeneous and pack controllers. The designer may decide that one of the controllers sufficiently satisfies the requirements for the task, or might wish to investigate other variations for a more suitable controller. The latter decision is facilitated by the ability to build behavior-based controllers that are easy to modify and evaluate in an expeditious manner, as we have demonstrated here.

A.7.5 Heterogeneity and Performance

So far, we have avoided a quantitative evaluation of the heterogeneity demonstrated by our three controllers. The reason for this is twofold:

1. Quantification of the heterogeneity of a multi-robot system can be subjective and ill-defined.
2. Regardless of how well-defined heterogeneity is, the link between it and performance may be unreliable.

We will consider each of these points in more depth.

Heterogeneity in multi-robot systems remains ill-defined partially because to date there has been little work exploring its quantification. One notable exception is work by Balch on *simple social entropy* and *hierarchical social entropy* (Balch 1997, Balch 2000). Both are based on information entropy (Shannon & Weaver 1963) and provide metrics for quantifying *behavioral* differences in a group of robots.

For illustrative purposes, we use simple social entropy which takes the form $\sum_{i=1}^{M} p_i \log_2(p_i)$, where M is the number of (behavioral) classes of robots, and p_i is the proportion of robots in the ith class. According to this measure, our homogeneous controller has one class containing all four robots, giving $p_1 = 1$ and a social entropy of 0, indicating homogeneity. The caste controller has two classes with $p_1 = 1/4 = .25$ and $p_2 = 3/4 = .75$, giving a social entropy of 0.81, indicating some heterogeneity. Though seemingly straightforward, calculating the social entropy of the pack controller introduces the dilemma of subjectivity. If we consider that all of the robots have the same controller and behave similarly, it seems clear that the group is homogeneous and has a social entropy of 0. If, however, we consider that each robot has a defined position in the hierarchy and behaves differently with respect to the other robots, then it appears that there are four classes, each containing one robot. This results in a social entropy of 2.0, indicating maximum heterogeneity. Is the pack controller fully homogeneous or heterogeneous? The fact that both views seem justified helps illustrate our first point: even with a well-defined metric, heterogeneity may still be subjective.

The situation is further complicated if the system contains multiple forms of heterogeneity. The caste controller, for instance, exhibits not only behavioral heterogeneity, but also spatial heterogeneity since the robots occupy different regions. We can quantify spatial heterogeneity using a variation of Balch's social entropy. The Search Caste contains 3 robots occupying 141 square feet (ft^2) of space, for a total of $3 \times 141 = 423$ robot-ft^2. The Goal Caste contains 1 robot and occupies 13 ft^2 of space, for a total of $1 \times 13 = 13$ robot-ft^2. In our calculation, $p_1 = 423/436 = 0.970$ and $p_2 = 13/436 = 0.030$, giving a *spatial* entropy of 0.19 and indicating a small amount of heterogeneity. The question is how to describe the overall heterogeneity of the caste controller. Should each type of heterogeneity (behavioral and spatial) be defined separately, or should the two numbers be combined into a single value? In the latter case, how should each number be weighted? The influence of each type of heterogeneity could depend on the task the robots are performing and the structure of the environment. Any weighting may thus have to be derived (likely empirically) for the exact scenario. If this is not possible, the overall heterogeneity of the system could remain ambiguous or ill-defined. The addition of other forms of heterogeneity (e.g., involving morphology or sensors) could further complicate the matter.

Even given an adequate measure for all forms of heterogeneity and their combination, the lack of a clear connection between the performance of the system and degree of heterogeneity it exhibits remains a concern. In our work in this appendix, we have compared several aspects of performance among our three controllers, including interference, time-to-completion, and energy expenditure. The important caveat is that these results are not completely general. They are partially dependent upon the structure of the environment, the physical characteristics of the robots, and the exact details of the controllers. In other words, the same task on different robots in a different environment might give very different results. Adding or removing heterogeneity to the system may improve or degrade performance depending on the details of the system and the aspect of heterogeneity being changed. As in the second point, one may not be able to rely on the results of a heterogeneity/performance comparison to generalize to another situation.

The quantified heterogeneity of a multi-robot system is a potentially important design and diagnosis parameter, but we have seen that it can be difficult to quantify, and once quantified, is of uncertain relation to performance. Based on our experimental results in foraging, it was not clear how this relationship could help the designer improve a multi-robot system. We therefore did not focus our analysis in this appendix upon this open research topic. Our hope, however, is that the capability to expeditiously build, modify, and evaluate multi-robot controllers, as we have demonstrated, will help facilitate the future study of issues in group robotics, such as the uses of a quantitative analysis of heterogeneity.

A.8 Summary

In this appendix, we have demonstrated the successful application of behavior-based control to the task of distributed multi-robot collection. Our focus has been on developing controllers that are robust to noise and robot failures, and easily modified to facilitate development of the variation that sufficiently satisfies the requirements for the task. Three versions of the collection task were presented: an initial homogeneous controller, and two heterogeneous variations (pack and caste) derived from the spatio-temporal manipulation of physical interference. All three versions were evaluated in a spatio-temporal context using interference, time-to-completion, and distance traveled as the main diagnostic parameters. This work demonstrates that given a good substrate for development (e.g., a useful set of behaviors), it can be relatively easy to implement, evaluate, and compare multi-robot controllers. As demonstrated in the body of this dissertation, such controllers can then be used in conjunction with AMMs to evaluate on-line, the robot-environment interaction dynamics for use in a variety of performance-improving applications.

Appendix B

Details of AMM Representation and Construction

This appendix details the AMM representation and model construction algorithm that are briefly presented in Chapter 4. The reader should refer to Chapter 4 for a discussion of the relationship between AMMs/MCs/SMPs and how AMMs are used with behavior-based control.

B.1 Representation of AMMs

The representation of a parametric AMM used by our model construction algorithm is sixteen elements ($\mathbf{S}$, $\mathbf{A}$, $\mathbf{B}$, $\mathbf{L}$, $\mathbf{T}$, $\mathbf{\Lambda}$, $\mathbf{\Upsilon}$, Alast, Llast, sym, oldsym, numsym, inlink, $\mathsf{outlink}$, $\mathsf{currnode}$, $\mathsf{oldnode}$) containing all of the information necessary for incremental model construction. The boldfaced elements are matrices or other compound structures, while the sans-serif-type elements are variables containing single numeric values. The details of the elements are as follows:

1. $\mathbf{S}$, a set of symbols $\{s_1, s_2, \ldots, s_M\}$ recognized by the network. The first symbol, s_1, is recognized only by the first state, a_1.

2. $\mathbf{A}$, a set of states (or nodes) $\{a_1, a_2, \ldots, a_N\}$. Each state a_i has four attributes:

   - a_i^s, the symbol that the state recognizes, i.e., an element of $\mathbf{S}$;
   - a_i^μ, the average number of time steps that the system remains in a_i whenever it enters that state;
   - $a_i^{\sigma^2}$, the variance associated with a_i^μ;
   - and a_i^p, the probability of remaining in a_i in the next time step.

   The state, a_1, represents the initial (unknown) state of the system, which is promptly left upon commencement of model construction and never entered subsequently.

3. $\mathbf{B}$, an $N \times M$ transition matrix, where $b_i(k)$ contains the value of the state to transition to if the current state is a_i and symbol s_k is observed. If $a_i^s = s_k$, then $b_i(k) = a_i$, i.e., if

the observed symbol is identical to the last symbol observed, then the system remains in the current state.

4. $\mathbf{L}$, a set of directed links $\{l_1, l_2, \dots, l_P\}$, connecting the states. Each link l_i has the following six attributes:

   - l_i^f, indicates the state <u>f</u>rom which the link begins, $a_{l_i^f} \in \mathbf{A}$;
   - l_i^t, indicates the state <u>t</u>o which the link connects, $a_{l_i^t} \in \mathbf{A}$. The following constraints apply: a link cannot start and end at the same state, $l_i^f \neq l_i^t$; and two links from the same state cannot go to states that accept the same symbol, $\forall i, j$ s.t. $l_i^f = l_j^f$, $a_{l_i^t}^s \neq a_{l_j^t}^s$;
   - l_i^δ, stores the number of times the link l_i has been traversed;
   - l_i^Σ, stores the total number of time steps that the system has been in state l_i^t, after first having traversed the link l_i;
   - $l_i^{\Sigma^2}$, contains the sum of squares of all the durations that comprise l_i^Σ;
   - and l_i^p is the probability of using the link l_i at each time step, given the system is in state l_i^f.

   Because no two links can have the same value for both their *from* and *to* attributes, they cannot represent the same directed transition. Thus, $N-1 \leq P \leq N(N-1)$: at least $N-1$ links are needed to connect the non-initial states, and for a fully connected network there are $N(N-1)$ links between the non-initial states. The single link from a_1, l_1, is traversed exactly one time, giving $l_1^\delta = 1$ and $l_1^p = 0$.

5. $\mathbf{T}$, a set of structures $\{\mathbf{T}^1, \dots, \mathbf{T}^{n_{\max}-1}\}$, each with elements $\{t_1^n, t_2^n, \dots, t_{Q_n}^n\}$ storing information on a particular n-link traversal sequence, where $1 < n \leq n_{\max} - 1$ and $n_{\max}$ is a user-specified maximum order for the model. Each element t_i^n has $n+4$ attributes:

   - $t_i^{n,1}, t_i^{n,2}, \dots, t_i^{n,n+1}$, the n links comprising t_i^n, stored as indices into $\mathbf{L}$;
   - $t_i^{n,\delta}$, the number of times the n-link sequence has been traversed;
   - $t_i^{n,\Sigma}$, the total number of time steps that the system has been in the state that link $t_i^{n,1}$ connects to, after first having traversed t_i^n;
   - t_i^{n,Σ^2}, the sum of the squares of all the durations that comprise $t_i^{n,\Sigma}$.

   The bounds of Q_n are given by:

$$\left.\begin{array}{ll} 0, & \text{if } P < n \\ \text{P-n+1}, & \text{if } P \geq 2 \end{array}\right\} \leq Q_n \leq N(N-1)^n.$$

   In order for a two-link transition to exist there must be at least two links. If more than two links exist, the fewest n-link transitions ($P-n+1$ of them) are created when an Euler path exists and is followed through the network. In a fully connected network, each of the

$P = N(N-1)$ links has a transition to $N-1$ other links, giving us the upper bound for a sequence of n links.

6. $\mathbf{\Lambda}$, a list of elements $\{\Lambda_1, \Lambda_2, \Lambda, \Lambda_{n_{\max}}\}$, indicating the $n_{\max}$ most recently used links in strict reverse chronological order, $l_{\Lambda_i} \in \mathbf{L}$, $1 \leq i \leq n_{\max}$, and Λ_i was used $i-1$ transitions in the past.

7. $\mathbf{\Upsilon}$, a list of elements $\{\Upsilon_1, \Upsilon_2, \ldots, \Upsilon_{n_{\max}-1}\}$, maintaining references to the penultimately used multi-link transition of each order, $t^i_{\Upsilon_i} \in \mathbf{T}^i$.

8. Alast, index of the last element of $\mathbf{A}$, a_{Alast}.

9. Llast, index of the last element of $\mathbf{L}$, l_{Llast}.

10. sym, the current input symbol to the model, $\mathsf{sym} \in \mathbf{S}$.

11. oldsym, the previous input symbol to the model, $\mathsf{oldsym} \in \mathbf{S}$.

12. numsym, the length of the uninterrupted sequence of identical input symbols beginning with sym; identical to the number of time steps spent in the current state.

13. inlink, a reference to the link used in transitioning *in*to the current state of the system.

14. $\mathsf{outlink}$, a reference to the link to be used in transitioning *out* of the current state.

15. $\mathsf{currnode}$, the current node (state) of the system.

16. $\mathsf{oldnode}$, the previous state of the system.

Given this AMM representation, the corresponding probabilistic transition matrix of a Markov chain could be generated from a^p, l^p, l^f, and l^t. The addition of a^μ and a^{σ^2} provides the more general state duration capabilities of an SMP. Aside from a^s, the remaining representational elements are used in incremental model generation and dynamic model reconfiguration using node splitting.

In addition to the sixteen elements above, the nonparametric representation for AMMs contains the following three elements:

1. $\mathcal{D}$, a finite set of elements $\{\mathcal{D}_1, \mathcal{D}_2, \ldots\}$ storing the input data. Each $\mathcal{D}_i$ stores information about the consecutive time spent in a particular state, and has two attributes:
   - $\mathcal{D}^a_i$, the state that the input data is associated with;
   - $\mathcal{D}^\tau_i$, the number of time steps spent in the state for the current entry into the state.

2. $\mathcal{D}^{\mathbf{L}}$, a finite set of elements $\{\mathcal{D}^{\mathbf{L}}_1, \ldots, \mathcal{D}^{\mathbf{L}}_P\}$, with each $\mathcal{D}^{\mathbf{L}}_i$ containing indices into $\mathcal{D}$ for the state given by l^t_i.

3. $\mathcal{D}^{\mathbf{T}}$, a finite set of elements $\{\mathcal{D}^{\mathbf{T}}_1, \ldots, \mathcal{D}^{\mathbf{T}}_{n_{\max}-1}\}$, with each $\mathcal{D}^{\mathbf{T}}_i$ containing indices $\{\mathcal{D}^{\mathbf{T}}_{i,1} \ldots \mathcal{D}^{\mathbf{T}}_{i,Q_i}\}$, and each $\mathcal{D}^{\mathbf{T}}_{i,j}$ an index into $\mathcal{D}$ for the state given by $t^{i,1}_j$.

The focus of this appendix is on parametric AMM construction, but the pseudo-code presented can easily be extended to accommodate nonparametric AMMs. Next we present the details of the AMM construction algorithm.

B.2 AMM Construction Algorithm

For simplicity, in the pseudo-code that follows, the representational elements of the AMM being constructed (as presented in the previous section) are considered global variables. In an attempt to keep the algorithm as concise as possible, we employ a mix of computational constructs and mathematical notation. Comments are provided in the hope of enhancing comprehensibility.

B.2.1 Initialization

The algorithm first initializes the elements of the AMM being constructed.

```
1.  $\mathbf{A} = \{a_1\}$;                                # the AMM starts with one state
2.  $a_1^s = 0$;                                         # the unique symbol for the first state is 0
3.  $a_1^\mu = 0$;                                       # mean time in the state is 0
4.  $a_1^{\sigma^2} = 0$;                                # sum squared durations is 0
5.  $a_1^p = 0$;                                         # probability of using the state is 0

6.  $\mathbf{B} = \{\}$;                                 # there are no transitions yet

7.  $\mathbf{L} = \{l_1\}$;                              # the AMM starts with one link
8.  $l_1^f = 0$;                                         # from some imaginary state
9.  $l_1^t = 1$;                                         # to $a_1$
10. $l_1^\delta = 1$;                                    # this link is used only once
11. $l_1^\Sigma = 1$;                                    # sum of durations in $a_1$ is 1
12. $l_1^{\Sigma^2} = 1$;                                # sum squared durations is 1
13. $l_1^p = 0$;                                         # probability of using this link again is 0

14. $\mathbf{T} = \{\mathbf{T}^1, \dots, \mathbf{T}^{n_{\max}-1}\}$;   # initialize $\mathbf{T}$
15. for $i = 1$ to $n_{\max} - 1$                        # initialize each $\mathbf{T}^i$
16.     $\mathbf{T}^i = \{\}$;                           # to contain no elements
17.     $Q_i = 0$;                                       # there are zero elements in $\mathbf{T}^i$
18. end

19. $\mathbf{S} = \emptyset$;                            # no symbols have been seen yet
```

```
20. Alast = 1;                                  # index of last state is 1
21. Llast = 1;                                  # index of last link is 1

22. sym = 0;                                    # unique symbol of the first state
23. oldsym = 0;                                 # initialize to anything
24. numsym = 1;                                 # one '0' has been observed

25. inlink = 1;                                 # we are using the first link
26. outlink = 1;                                # initialize to anything

27. currnode = 1;                               # current state is 1
28. oldnode = 0;                                # old state is 0 (imaginary state)

29. $\boldsymbol{\Upsilon} = \{\Upsilon_1, \ldots, \Upsilon_{n_{\max}-1}\}$;    # initialize $\boldsymbol{\Upsilon}$
30. for $i = 1$ to $n_{\max} - 1$
31.     $\Upsilon_i = 0$;                           # to be all zeros
32. end

33. $\boldsymbol{\Lambda} = \{\Lambda_1, \ldots, \Lambda_{n_{\max}}\}$;        # initialize $\boldsymbol{\Lambda}$
34. for $i = 1$ to $n_{\max} - 2$
35.     $\Lambda_i = 0$;                            # to be all zeros
36. end
37. $\Lambda_1$ = outlink;                        # most recently used link
38. $\Lambda_2$ = outlink;                        # next most recently used link
```

B.2.2 Main Loop

Below we present the pseudo-code for the main loop of the algorithm that is executed for every input symbol. At the start of the code, sym holds the current input symbol that is to be incorporated into the model, and oldsym holds the last input symbol. As mentioned previously, numsym contains the number of symbols observed in the current state, oldnode stores the index of the last state the system was in, currnode stores the index of the current state, inlink holds the index of the link traversed to enter the current state, and outlink holds the index of the link to be traversed in leaving the current state.

```
1. if (oldsym == sym)                  # if the system remains in the same state
2.     numsym += 1;                    # increment the time spent in the state
3.     $l^{\Sigma}_{\text{inlink}}$ += 1;                 # increment time spent in the state
                                       #     that $l_{\text{inlink}}$ connects to
```

```
4.        $i = 1;$                                      # a counter
5.        while $(i < n_{\max}$ & $\Upsilon_i \neq 0)$          # update $\boldsymbol{\Upsilon}$
6.            $t^{i,\Sigma}_{\Upsilon_i}$ += 1;                       # increment the time spent in the state
                                                        #    that $t^{i}_{\Upsilon_i}$ ends at
7.            $i$ += 1;                                   # increment the counter
8.        end
9.        traversal_prob(currnode);                     # update the transition probabilities
10.   else                                              # the system is transitioning to a new state
11.       $l^{\Sigma^2}_{\mathsf{inlink}}$ += numsym$^2$;                    # increment the time spent in the state
                                                        #    that $l_{\mathsf{inlink}}$ connects to
12.       node_prob(currnode);                          # update mean/variance for the current state
13.       $i = 1;$
14.       while $(i < n_{\max}$ & $\Upsilon_i \neq 0)$          # update $\boldsymbol{\Upsilon}$
15.           $t^{i,\Sigma^2}_{\Upsilon_i}$ += numsym$^2$;                  # increment the sum squared time spent
                                                        #    in the state that $t^{i}_{\Upsilon_i}$ ends at
16.           $i$ += 1;
17.       end
18.       if (sym $\notin \mathbf{S}$)                              # if the symbol has not been seen before
19.           $\mathbf{S} = \mathbf{S} \cup \{$sym$\}$;                       # add the symbol to $\mathbf{S}$
20.           Alast += 1;                               # add a new state
21.           $a^{s}_{\mathsf{Alast}}$ = sym;                           # the new state recognizes the symbol
22.           for $i$ = 1 to Alast
23.               $b_i$(sym) = Alast;                       # create the new transitions for sym
24.           end
25.           $\forall i$ s.t. $i \in \mathbf{S}$, let $b_{\mathsf{Alast}}(i) = b_1(i)$# initialize transitions for the new state
26.       end
27.       oldnode = currnode;
28.       currnode = $b_{\mathsf{oldnode}}$(sym);                    # transition to the new state
29.       outlink = $i$ s.t. ($l^{f}_{i}$ == oldnode & $l^{t}_{i}$ == currnode)# find the link to transition on
30.       if ($\neg\exists$ outlink)                              # must create a new link
31.           Llast += 1;                               # create the new link
32.           $l^{f}_{\mathsf{Llast}}$ = oldnode;                        #link connects from oldnode
33.           $l^{t}_{\mathsf{Llast}}$ = currnode;                       #link connects to currnode
34.           outlink = Llast;                          # update outlink
35.       end
36.       $\Lambda_{2:n_{\max}} = \Lambda_{1:n_{\max}-1}$;                       # shift the history of used links
37.       $\Lambda_1$ = outlink;                              # add the new link to the front of the history
38.       numsym = 1;                                   # reset the time spent in the current state
39.       $l^{\delta}_{\mathsf{outlink}}$ += 1;                             # increment number of times link has
```

```
                                                  #    been traversed
40.      $l^{\Sigma}_{\mathsf{outlink}}$ += 1;                  # increment the time spent in the state
                                                  #    that $l_{\mathsf{outlink}}$ connects to
41.      traversal_prob(oldnode);                 # update the transition probabilities
42.      $i = 1$;
43.      while ($i < n_{\max}$ & $\Lambda_{i+1} \neq 0$)          # update $\boldsymbol{\Upsilon}$
44.          $j = k$  s.t.  $t^{i,1:i+1}_k = \Lambda_{1:i+1}$;      # for every order find the $t^i$ traversed
45.          if ($\neg\exists\ j$)                       # if the $t^i$
46.              $Q_i$ += 1;                         # create a new $t^i$
47.              $t^{i,1:i+1}_{Q_i} = \Lambda_{1:i+1}$;            # initialize it
48.              $t^{i,\delta}_{Q_i} = 0$;
49.              $t^{i,\Sigma}_{Q_i} = 0$;
50.              $t^{i,\Sigma^2}_{Q_i} = 0$;
51.              $j = Q_i$;
52.          end;
53.          $t^{i,\delta}_j$ += 1;                         # increment number of times traversed
54.          $t^{i,\Sigma}_j$ += 1;                         # increment time spent in the state it connects to
55.          $\Upsilon_i = j$;                          # update the must recently used traversals
56.      end
57.      if ($n_{\max} > 1$)                          # if user-specified order > 1
58.          do_node_split();                     # test to see if node-splitting is needed
59.      end
60.      inlink = outlink;                        # the last step in transitioning to the new state
61.  end
```

B.2.3 Calculating Traversal Probabilities

The following pseudo-code function calculates the traversal probabilities associated with a particular node and updates the appropriate statistics in the node and its links.

```
function traversal_prob(node)
1.   $x_1 = \{i \mid l^t_i == \mathsf{node}\}$;                  # find all incoming links to node
2.   $n_1 = \sum_{\text{all } i \in x_1} (l^{\Sigma}_i - l^{\delta}_i)$;              # total self-transitions of node
3.   $x_2 = \{i \mid l^f_i == \mathsf{node}\}$;                  # find all outgoing links from node
4.   $n_2 = \sum_{\text{all } i \in x_2} l^{\delta}_i$;                      # total times node has been entered
5.   if ($n_1 + n_2 \neq 0$)
6.       $a^p_{\mathsf{node}} = \frac{n_1}{n_1+n_2}$;                    # calculate the probability of a self-transition
```

```
7.      $l^p_{x_2} = \frac{l^\delta_{x_2}}{n_1+n_2}$;                 # calculate outgoing transition probabilities
8.  end
```

B.2.4 Calculating Node Probabilities

The following function updates the mean and variance for the particular node passed as a parameter.

```
function node_prob(node)
1.  $x_1 = \{i \mid l^t_i == \mathsf{node}\}$;               # find all incoming links to node
2.  $n_1 = \sum_{\text{all } i \in x_1} l^\delta_i$;          # total times node has been entered
3.  $n_2 = \sum_{\text{all } i \in x_1} l^\Sigma_i$;          # total time spent in node
4.  if $(n_1 > 0)$                                  # update mean time spent in node
5.      $a^\mu_{\mathsf{node}} = n_2/n_1$;
6.  else
7.      $a^\mu_{\mathsf{node}} = 0$;
8.  end
9.  $n_3 = \sum_{\text{all } i \in x_1} l^{\Sigma^2}_i$;      # total of squared durations spent in node
10. if $(n_1 > 1)$                                  # update the variance associated with node
11.     $a^{\sigma^2}_{\mathsf{node}} = (n_3 - 2 * a^\mu_{\mathsf{node}} * n_2 + n1 * [a^\mu_{\mathsf{node}}]^2)/(n_1 - 1)$;
12. else
13.     $a^{\sigma^2}_{\mathsf{node}} = 0$;
14. end
```

B.2.5 Node Splitting

This function determines whether node splitting is necessary, and if it is, splits the nodes and creates new links and mutli-link transitions as appropriate.

```
function do_node_split()
1.  flag = 0;                                   # boolean indicating if splitting is necessary
2.  splitorder = 1;                             # current Markovian order checked for splitting
3.  while ((splitorder < $n_{\max}$) & (flag == 0)) # check all orders
4.      splitorder += 1;
5.      $x_1 = \{i \mid l^f_i == \mathsf{oldnode}\}$;       # find out-links
```

```
6.    $\forall i \mid i \in x_1,\, p_i = l_i^p$;                      # get link probabilities
7.    $\forall i \mid i \in x_1,\, p_i = p_i / \left( \sum_{\text{all } i \in x_1} p_i \right)$;       # normalize probabilities
8.    $\mathsf{flag} = 0$;
9.    $x_2 = \{k \mid t_k^{[\mathsf{splitorder}-1, 2:\mathsf{splitorder}]} == \Lambda_{2:\mathsf{splitorder}}\}$;   # find multi-link traversals
10.   if ($\mathsf{splitorder} == 2$)                     # get total transitions
11.       $\mathsf{transitions} = l_{\mathsf{inlink}}^{\delta}$;
12.   else
13.       $\mathsf{transitions} = \sum_{\text{all i} \in \text{x}_2} t_{x_{2,j}}^{[\mathsf{splitorder}-1, \mathsf{splitorder}+1]}$;
14.   end
15.   for $j = 1$ to $length(x_2)$
16.       if ($\mathsf{transitions} > 0$)
              # calculate binomial limits as in Section 3.3.1
              # with $x = t_{x_{2,j}}^{[\mathsf{splitorder}-1, \mathsf{splitorder}+1]}$, $n = \mathsf{transitions}$,
              # and at a significance level of $\alpha = 0.05$ or $0.01$
17.           $\mathsf{lcl}$ = lower binomial confidence limit;
18.           $\mathsf{ucl}$ = upper binomial confidence limit;
19.       else
20.           $\mathsf{lcl} = 0$;
21.           $\mathsf{ucl} = 1$;
22.       end
23.       if $\left( p_{t_{x_{2,j}}^{[\mathsf{splitorder}-1,1]}} < \mathsf{lcl} \mid p_{t_{x_{2,j}}^{[\mathsf{splitorder}-1,1]}} > \mathsf{ucl} \right)$
24.           $\mathsf{flag} = 1$;                         # do node-splitting
25.       end
26.   end
27.   if ($\mathsf{flag} == 0$)
28.       if (test_node_durations() == 1)
              # if there are inconsistencies in the time spent
              # in the state that is being left, then split
29.           $\mathsf{flag} = 1$;                         # do node-splitting
30.       end
31.   end
32.   if (($\mathsf{flag} == 1$) & (all elements of $\Lambda_{1:\mathsf{splitorder}}$ are unique))  # split states!
33.       $\mathsf{Alast} = \mathsf{Alast} + 1$;                 # add a new state
34.       $a_{\mathsf{Alast}}^s = a_{l_{\Lambda_{\mathsf{splitorder}}}^t}^s$;                 # set the symbol
35.       $\forall i \mid s_i \in S,\, b_{\mathsf{Alast}}(i) = b_{l_{\Lambda_{\mathsf{splitorder}}}^t}(i)$;  # adjust $\mathbf{B}$
36.       $l_{\Lambda_{\mathsf{splitorder}}}^t = \mathsf{Alast}$;                  # move in-link
37.       $b_{l_{\Lambda_{\mathsf{splitorder}}}^f}(a_{\mathsf{Alast}}^s) = \mathsf{Alast}$;
```

```
38.     $b_{\mathsf{Alast}}(a^s_{\mathsf{Alast}}) = \mathsf{Alast}$;
39.     $\mathbf{\Lambda}^{'} = \mathbf{\Lambda}$;                    # make a temporary list of links
        # make the rest of the new states with new links between them
40.     for $i = \mathsf{splitorder} - 1$ downto 2
41.         $\mathsf{Alast} = \mathsf{Alast} + 1$;
42.         $a^s_{\mathsf{Alast}} = a^s_{l^t_{\Lambda_i}}$;
43.         $\forall i \mid s_i \in S,\ b_{\mathsf{Alast}}(i) = b_{l^t_{\Lambda_i}}(i)$;
44.         $\mathsf{Llast} = \mathsf{Llast} + 1$;
45.         $l^f_{\mathsf{Llast}} = \mathsf{Alast} - 1$;
46.         $l^t_{\mathsf{Llast}} = \mathsf{Alast}$;
            # find the multi-link traversal
47.         $\mathsf{temp} = k$ s.t. $t_k^{[\mathsf{splitorder}-i,1:\mathsf{splitorder}-i+1]} == \Lambda_{i:\mathsf{splitorder}}$;
48.         $l^\delta_{\mathsf{Llast}} = t_{\mathsf{temp}}^{[\mathsf{splitorder}-i,\delta]}$;
49.         $l^\delta_{\Lambda_i} = l^\delta_{\Lambda_i} - l^\delta_{\mathsf{Llast}}$;
50.         $l^\Sigma_{\mathsf{Llast}} = t_{\mathsf{temp}}^{[\mathsf{splitorder}-i,\Sigma]}$;
51.         $l^\Sigma_{\Lambda_i} = l^\Sigma_{\Lambda_i} - l^\Sigma_{\mathsf{Llast}}$;
52.         $l^{\Sigma^2}_{\mathsf{Llast}} = t_{\mathsf{temp}}^{[\mathsf{splitorder}-i,\Sigma^2]}$;
53.         $l^{\Sigma^2}_{\Lambda_i} = l^{\Sigma^2}_{\Lambda_i} - l^{\Sigma^2}_{\mathsf{Llast}}$;
54.         $\Lambda^{'}_i = \mathsf{Llast}$;
55.         $t_{\mathsf{temp}}^{[\mathsf{splitorder}-i,1:\mathsf{splitorder}-i+1]} = \Lambda^{'}_{i:\mathsf{splitorder}}$;
56.         $b_{l^f_{\mathsf{Llast}}}(a^s_{\mathsf{Alast}}) = \mathsf{Alast}$;
57.         $b_{\mathsf{Alast}}(a^s_{\mathsf{Alast}}) = \mathsf{Alast}$;
58.     end
59.     for $i = \mathsf{splitorder} - 1$ downto 2
60.         for $k = \mathsf{splitorder} - i + 1$ to $n_{\max} - 1$
61.             $\mathsf{temp} = \{j \mid t_j^{[k,1:\mathsf{splitorder}-i+1]} == \Lambda_{i:\mathsf{splitorder}}\}$;
62.             $\forall j \in \mathsf{temp}, t_j^{[k,1:\mathsf{splitorder}-i+1]} = \Lambda^{'}_{i:\mathsf{splitorder}}$;
63.         end
64.     end
65.     for $k_1 = 2$ to $\mathsf{splitorder} - 1$
66.         for $k_2 = k_1 + 1$ to $\mathsf{splitorder} - 1$
67.             $\mathsf{temp} = j$ s.t. $t_j^{[k_2-k_1,k_2-k_1+1]} = \Lambda_{k_1:k_2}$;
68.             $Q_{k_2-k_1}$ += 1;                 # add another element to $t$
69.             $t_{Q_{k_2-k_1}}^{[k_2-k_1,1:k_2-k_1+1]} = \Lambda^{'}_{k_1:k_2}$;
70.             $t_{Q_{k_2-k_1}}^{[k_2-k_1,\delta]} = l^\delta_{\Lambda^{'}_{k_1}}$;
71.             $t_{\mathsf{temp}}^{[k_2-k_1,\delta]}$ -= $l^\delta_{\Lambda^{'}_{k_1}}$;
72.             $t_{Q_{k_2-k_1}}^{[k_2-k_1,\Sigma]} = l^\Sigma_{\Lambda^{'}_{k_1}}$;
73.             $t_{\mathsf{temp}}^{[k_2-k_1,\Sigma]}$ -= $l^\Sigma_{\Lambda^{'}_{k_1}}$;
```

```
74.         $t^{[k_2-k_1,\Sigma^2]}_{Q_{k_2-k_1}} = l^{\Sigma^2}_{\Lambda'_{k_1}}$;
75.         $t^{[k_2-k_1,\Sigma^2]}_{\mathsf{temp}} -= l^{\Sigma^2}_{\Lambda'_{k_1}}$;
76.     end
77. end
78. for $\mathsf{linkloop} = \mathsf{splitorder}$ downto 2
79.     $x_2 = \{j \mid t_j^{[\mathsf{splitorder}-\mathsf{linkloop}+1,2:\mathsf{splitorder}-\mathsf{linkloop}+2]} == \Lambda_{\mathsf{linkloop}:\mathsf{splitorder}}\}$;
80.     for $j = 1$ to $\mathit{length}(x_2)$
81.         if $\neg((t_{x_{2,j}}^{[\mathsf{splitorder}-\mathsf{linkloop}+1,1]} == \Lambda'_{\mathsf{linkloop}-1})\&(\mathsf{linkloop} > 2))$
82.             $\mathsf{Llast} += 1$;
83.             $l^f_{\mathsf{Llast}} = l^t_{\Lambda'_{\mathsf{linkloop}}}$;
84.             $n = t_{x_{2,j}}^{[\mathsf{splitorder}-\mathsf{linkloop}+1,1]}$;
85.             $\mathsf{transitions} = l^\delta_n$;
86.             $l^t_{\mathsf{Llast}} = l^t_n$;
87.             $l^\delta_{\mathsf{Llast}} = t_{x_{2,j}}^{[\mathsf{splitorder}-\mathsf{linkloop}+1,\delta]}$;
88.             $l^\delta_n -= l^\delta_{\mathsf{Llast}}$;
89.             $l^\Sigma_{\mathsf{Llast}} = t_{x_{2,j}}^{[\mathsf{splitorder}-\mathsf{linkloop}+1,\Sigma]}$;
90.             $l^\Sigma_n -= l^\Sigma_{\mathsf{Llast}}$;
91.             $l^{\Sigma^2}_{\mathsf{Llast}} = t_{x_{2,j}}^{[\mathsf{splitorder}-\mathsf{linkloop}+1,\Sigma^2]}$;
92.             $l^{\Sigma^2}_n -= l^{\Sigma^2}_{\mathsf{Llast}}$;
93.             for $k_1 = \mathsf{splitorder} - \mathsf{linkloop}$ downto 0
94.                 if $(k_1 > 0)$
95.                     $x_3 = \mathsf{temp}$  s.t.  $t_{\mathsf{temp}}^{[k_1,1:k_1+1]} == [n, \Lambda_{\mathsf{linkloop}:\mathsf{linkloop}+k_1-1}]$;
96.                     $\mathsf{temp} = j \mid t_j^{[\mathsf{splitorder}-\mathsf{linkloop}+1,1:\mathsf{splitorder}-\mathsf{linkloop}+2]}$
97.                             $== \left[t_{x_3}^{[k_1,1:k_1+1]}, \Lambda_{\mathsf{linkloop}+k_1:\mathsf{splitorder}}\right]$;
98.                     $Q_{k_1} += 1$;
99.                     $t_{Q_{k_1}}^{[k_1,1:k_1+1]} = \left[\mathsf{Llast}, \Lambda'_{\mathsf{linkloop}:\mathsf{linkloop}+k_1-1}\right]$;
100.                    $t_{Q_{k_1}}^{[k_1,\delta]} = t_{\mathsf{temp}}^{[\mathsf{splitorder}-\mathsf{linkloop}+1,\delta]}$;
101.                    $t_{x_3}^{[k_1,\delta]} -= t_{Q_{k_1}}^{[k_1,\delta]}$;
102.                    $t_{Q_{k_1}}^{[k_1,\Sigma]} = t_{\mathsf{temp}}^{[\mathsf{splitorder}-\mathsf{linkloop}+1,\Sigma]}$;
103.                    $t_{x_3}^{[k_1,\Sigma]} -= t_{Q_{k_1}}^{[k_1,\Sigma]}$;
104.                    $t_{Q_{k_1}}^{[k_1,\Sigma^2]} = t_{\mathsf{temp}}^{[\mathsf{splitorder}-\mathsf{linkloop}+1,\Sigma^2]}$;
105.                    $t_{x_3}^{[k_1,\Sigma^2]} -= t_{Q_{k_1}}^{[k_1,\Sigma^2]}$;
106.                end
107.                for $k_2 = k_1 + 1$ to $n_{\max} - 1$
108.                    $x_3 = \{j \mid t_j^{[k_2,k_2+1-k_1:k_2+1]} == [n, \Lambda_{\mathsf{linkloop}:\mathsf{linkloop}+k_1-1}]\}$;
109.                    $c = k_2 + \mathsf{splitorder} - (\mathsf{linkloop} + k_1 - 1)$;
110.                    if $(c + 1 \le n_{\max})$
```

```
111.             for $k_3 = 1$ to $length(x_3)$
112.                 $\mathsf{temp} = j \quad \text{s.t.} \quad t_j^{[c,1:c+1]}$
                         $== \left[t_{x_{3,k_3}}^{[k_2,1:k_2+1]}, \Lambda_{\mathsf{linkloop}+k_1:\mathsf{splitorder}}\right];$
113.                 if $(\exists \mathsf{temp})$
114.                     $Q_{k_2} \mathrel{+}= 1;$
115.                     $t_{Q_{k_2}}^{[k_2,1:k_2-k_1]} = t_{x_{3,k_3}}^{[k_2,1:k_2-k_1]};$
116.                     $t_{Q_{k_2}}^{[k_2,k_2+1-k_1:k_2+1]} = \left[\mathsf{Llast}, \Lambda'_{\mathsf{linkloop}:\mathsf{linkloop}+k_1-1}\right];$
117.                     $t_{Q_{k_2}}^{[k_2,\delta]} = t_{\mathsf{temp}}^{[c,\delta]};$
118.                     $t_{x_{3,k_3}}^{[k_2,\delta]} \mathrel{-}= t_{Q_{k_2}}^{[k_2,\delta]};$
119.                     $t_{Q_{k_2}}^{[k_2,\Sigma]} = t_{\mathsf{temp}}^{[c,\Sigma]};$
120.                     $t_{x_{3,k_3}}^{[k_2,\Sigma]} \mathrel{-}= t_{Q_{k_2}}^{[k_2,\Sigma]};$
121.                     $t_{Q_{k_2}}^{\left[k_2,\Sigma^2\right]} = t_{\mathsf{temp}}^{\left[c,\Sigma^2\right]};$
122.                     $t_{x_{3,k_3}}^{\left[k_2,\Sigma^2\right]} \mathrel{-}= t_{Q_{k_2}}^{\left[k_2,\Sigma^2\right]};$
123.                 end
124.             end
125.         else
126.             for $k_3 = 1$ to $length(x_3)$
127.                 $p_3 = l_{\mathsf{Llast}}^{\delta} / \mathsf{transitions};$
128.                 if $(p_3 > 0)$
129.                     $Q_{k_2} \mathrel{+}= 1;$
130.                     $t_{Q_{k_2}}^{[k_2,1:k_2-k_1]} = t_{x_{3,k_3}}^{[k_2,1:k_2-k_1]};$
131.                     $t_{Q_{k_2}}^{[k_2,k_2+1-k_1:k_2+1]} = \left[\mathsf{Llast}, \Lambda'_{\mathsf{linkloop}:\mathsf{linkloop}+k_1-1}\right];$
132.                     $t_{Q_{k_2}}^{[k_2,\delta]} = round\left(p_3 \cdot t_{x_{3,k_3}}^{[k_2,\delta]}\right);$
133.                     $t_{x_{3,k_3}}^{[k_2,\delta]} \mathrel{-}= t_{Q_{k_2}}^{[k_2,\delta]};$
134.                     $t_{Q_{k_2}}^{[k_2,\Sigma]} = round\left(p_3 \cdot t_{x_{3,k_3}}^{[k_2,\Sigma]}\right);$
135.                     $t_{x_{3,k_3}}^{[k_2,\Sigma]} \mathrel{-}= t_{Q_{k_2}}^{[k_2,\Sigma]};$
136.                     $t_{Q_{k_2}}^{\left[k_2,\Sigma^2\right]} = round\left(p_3 \cdot t_{x_{3,k_3}}^{\left[k_2,\Sigma^2\right]}\right);$
137.                     $t_{x_{3,k_3}}^{\left[k_2,\Sigma^2\right]} \mathrel{-}= t_{Q_{k_2}}^{\left[k_2,\Sigma^2\right]};$
138.                 end
139.             end
140.         end
141.     end
142. end
143. for $k_1 = \mathsf{splitorder} - \mathsf{linkloop} + 1$ to $n_{\max} - 1$
144.     for $k_2 = 1$ to $k_1 - \mathsf{splitorder} + \mathsf{linkloop}$
145.         $\mathsf{temp} = \{j \mid t_j^{[k_1,k_2:\mathsf{splitorder}-\mathsf{linkloop}+k_2+1]} == [n, \Lambda_{\mathsf{linkloop}:\mathsf{splitorder}}]\};$
146.         $\forall j \in \mathsf{temp}, t_j^{[k_1,k_2:\mathsf{splitorder}-\mathsf{linkloop}+k_2+1]} = \left[\mathsf{Llast}, \Lambda'_{\mathsf{linkloop}:\mathsf{splitorder}}\right];$
```

```
            end
          end
        end
        for $n_2 = 1$ to $\mathsf{Alast}$
          node_prob($n_2$);
          traversal_prob($n_2$);
        end
      end
    end
    $x_1 = j \quad \text{s.t.} \quad (l_j^f == \mathsf{Alast})\&(l_j^t == b_{\mathsf{Alast}}(\mathsf{sym}))$;
    $\Lambda_1^{'} = x_1$;
    $\mathbf{\Lambda} = \mathbf{\Lambda}^{'}$;
    $\mathsf{outlink} = x_1$;
    $\mathsf{oldnode} = \mathsf{Alast}$;
    $\mathsf{currnode} = l_{x_1}^t$;
    $i = 1$;
    reinitialize $x$;
    while $(i < n_{\max})$ & $(\Lambda_{i+1} \neq 0)$
      $\mathsf{temp} = j \quad \text{s.t.} \quad t_j^{[i,1:i+1]} == \Lambda_{1:i+1}$;
      if $(\neg\exists \mathsf{temp})$
        $x_i = 0$;
      else
        $x_i = \mathsf{temp}$;
      end
      $i \mathrel{+}= 1$;
    end
    $\Upsilon_{1:length(\mathrm{x})} = x$;
  end
end
```

B.3 Summary

This appendix presented many of the details of the AMM representation and pseudo-code for the model construction algorithm used in this dissertation.

Appendix C

Tables of Critical Points for $\hat{T}$

This appendix provides tables of critical points for the nonparametric test of location by Fligner & Rust (1982), which is described in detail in Section 3.4.1 of this dissertation.

The $\hat{T}$ statistic by Fligner & Rust (1982), a modification of Mood's (1954) statistic, enables a nonparametric median test that makes very few distribution assumptions. In particular, it accommodates non-symmetric and non-identically shaped distributions. This dissertation uses the $\hat{T}$-test in the construction of nonparametric AMMs (Appendix B) and in testing the significance of the experimental data in Chapter 8. Unfortunately, there is no convenient source for the critical points of $\hat{T}$, making its use impractical. This appendix attempts to ameliorate this situation by presenting tables of critical points for sample sizes $3 \leq m, n \leq 25$, which were calculated for the work in this dissertation. The values for each pair of sample sizes were derived using a $100,000$-iteration Monte Carlo simulation. Because the distribution of $\hat{T}$ is symmetric, and thus, in order to avoid redundancy, only the upper half (i.e., with $\min(m, n)$ indexing the row) of the full 23×23 table is presented in this appendix. When $m, n > 25$, the inverse cumulative normal distribution provides a good approximation to the critical points of $\hat{T}$ (see Section 3.4.1).

The tables of this appendix are interpreted in the following manner. Each 7×3 cell of a table is indexed by a particular combination of sample sizes, $\{\min(m, n), \max(m, n)\}$. The first column of each cell gives the nominal significance level, α, for the critical point, and the second column provides the actual significance level. The third column provides $\hat{T}_\alpha\{\min(m, n), \max(m, n)\}$, the critical point for the $\hat{T}$ distribution at a nominal significance level of α.

m, n	3			4			5			6		
3	0.1000	0.0497	2.4490	0.1000	0.0864	1.5000	0.1000	0.0716	2.0990	0.1000	0.0827	1.4020
	0.0500	0.0497	2.4490	0.0500	0.0289	2.2500	0.0500	0.0716	2.0990	0.0500	0.0470	1.9990
	0.0250	0.0497	2.4490	0.0250	0.0289	2.2500	0.0250	0.0181	2.1900	0.0250	0.0120	2.1040
	0.0100	0.0497	2.4490	0.0100	0.0289	2.2500	0.0100	0.0181	2.1900	0.0100	0.0120	2.1040
	0.0050	0.0497	2.4490	0.0050	0.0289	2.2500	0.0050	0.0181	2.1900	0.0050	0.0120	2.1040
	0.0025	0.0497	2.4490	0.0025	0.0289	2.2500	0.0025	0.0181	2.1900	0.0025	0.0120	2.1040
	0.0010	0.0497	2.4490	0.0010	0.0289	2.2500	0.0010	0.0181	2.1900	0.0010	0.0120	2.1040
4				0.1000	0.1575	1.4140	0.1000	0.1197	1.3280	0.1000	0.1182	1.2900
				0.0500	0.0137	2.8280	0.0500	0.0388	1.9910	0.0500	0.0237	2.4680
				0.0250	0.0137	2.8280	0.0250	0.0314	1.9930	0.0250	0.0237	2.4680
				0.0100	0.0137	2.8280	0.0100	0.0072	2.6570	0.0100	0.0094	2.5810
				0.0050	0.0137	2.8280	0.0050	0.0072	2.6570	0.0050	0.0094	2.5810
				0.0025	0.0137	2.8280	0.0025	0.0072	2.6570	0.0025	0.0094	2.5810
				0.0010	0.0137	2.8280	0.0010	0.0072	2.6570	0.0010	0.0094	2.5810
5							0.1000	0.1045	1.8000	0.1000	0.1115	1.1990
							0.0500	0.0563	1.8970	0.0500	0.0539	1.7980
							0.0250	0.0042	3.1620	0.0250	0.0134	2.3970
							0.0100	0.0042	3.1620	0.0100	0.0091	2.3980
							0.0050	0.0042	3.1620	0.0050	0.0022	2.9980
							0.0025	0.0042	3.1620	0.0025	0.0022	2.9980
							0.0010	0.0042	3.1620	0.0010	0.0022	2.9980
6										0.1000	0.1472	1.1540
										0.0500	0.0398	2.1900
										0.0250	0.0204	2.3090
										0.0100	0.0009	3.4640
										0.0050	0.0009	3.4640
										0.0025	0.0009	3.4640
										0.0010	0.0009	3.4640

Table C.1: Critical points, $\hat{T}_\alpha\{3\ldots6, 3\ldots6\}$, for the $\hat{T}$ distribution.

m, n	7			8			9			10		
	0.1000	0.0825	1.8640	0.1000	0.1161	1.2890	0.1000	0.0904	1.7320	0.1000	0.1102	1.2630
	0.0500	0.0578	1.9980	0.0500	0.0431	1.9340	0.0500	0.0452	1.9410	0.0500	0.0352	1.8940
	0.0250	0.0083	2.0700	0.0250	0.0431	1.9340	0.0250	0.0452	1.9410	0.0250	0.0352	1.8940
3	0.0100	0.0083	2.0700	0.0100	0.0059	2.0160	0.0100	0.0046	1.9990	0.0100	0.0038	1.9620
	0.0050	0.0083	2.0700	0.0050	0.0059	2.0160	0.0050	0.0046	1.9990	0.0050	0.0038	1.9620
	0.0025	0.0083	2.0700	0.0025	0.0059	2.0160	0.0025	0.0046	1.9990	0.0025	0.0038	1.9620
	0.0010	0.0083	2.0700	0.0010	0.0059	2.0160	0.0010	0.0046	1.9990	0.0010	0.0038	1.9620
	0.1000	0.1070	1.2420	0.1000	0.0913	1.2240	0.1000	0.0947	1.1900	0.1000	0.1354	1.1590
	0.0500	0.0460	1.7630	0.0500	0.0311	2.1780	0.0500	0.0488	1.6010	0.0500	0.0509	1.1830
	0.0250	0.0339	1.8640	0.0250	0.0248	2.3570	0.0250	0.0210	2.1350	0.0250	0.0312	2.2170
4	0.0100	0.0063	2.4850	0.0100	0.0061	2.4490	0.0100	0.0043	2.3850	0.0100	0.0135	2.3180
	0.0050	0.0063	2.4850	0.0050	0.0061	2.4490	0.0050	0.0043	2.3850	0.0050	0.0010	2.3660
	0.0025	0.0063	2.4850	0.0025	0.0061	2.4490	0.0025	0.0043	2.3850	0.0025	0.0010	2.3660
	0.0010	0.0063	2.4850	0.0010	0.0061	2.4490	0.0010	0.0043	2.3850	0.0010	0.0010	2.3660
	0.1000	0.0983	1.6770	0.1000	0.0873	1.4680	0.1000	0.0930	1.6200	0.1000	0.0996	1.3410
	0.0500	0.0531	1.7560	0.0500	0.0510	1.6920	0.0500	0.0360	1.6730	0.0500	0.0458	1.6260
	0.0250	0.0079	2.7950	0.0250	0.0162	2.1320	0.0250	0.0360	1.6730	0.0250	0.0188	1.9240
5	0.0100	0.0079	2.7950	0.0100	0.0114	2.2590	0.0100	0.0107	2.5710	0.0100	0.0126	2.1730
	0.0050	0.0040	2.9270	0.0050	0.0046	2.6650	0.0050	0.0077	2.7210	0.0050	0.0058	2.5810
	0.0025	0.0040	2.9270	0.0025	0.0024	2.8240	0.0025	0.0013	2.7880	0.0025	0.0017	2.7160
	0.0010	0.0040	2.9270	0.0010	0.0024	2.8240	0.0010	0.0013	2.7880	0.0010	0.0017	2.7160
	0.1000	0.0773	1.5260	0.1000	0.1367	1.0800	0.1000	0.0832	1.3480	0.1000	0.1183	1.0320
	0.0500	0.0665	1.6520	0.0500	0.0513	1.9610	0.0500	0.0562	1.5660	0.0500	0.0570	1.8970
	0.0250	0.0250	2.0560	0.0250	0.0215	2.1600	0.0250	0.0211	2.0860	0.0250	0.0174	2.0650
6	0.0100	0.0144	2.2030	0.0100	0.0026	3.1510	0.0100	0.0055	2.4570	0.0100	0.0035	2.8460
	0.0050	0.0041	2.7530	0.0050	0.0026	3.1510	0.0050	0.0050	2.6080	0.0050	0.0035	2.8460
	0.0025	0.0024	2.7540	0.0025	0.0026	3.1510	0.0025	0.0014	2.9490	0.0025	0.0028	3.0200
	0.0010	0.0006	3.3050	0.0010	0.0011	3.2400	0.0010	0.0008	3.1330	0.0010	0.0007	3.0980
	0.1000	0.1303	1.5550	0.1000	0.0997	1.2830	0.1000	0.1161	1.4690	0.1000	0.1045	1.4120
	0.0500	0.0693	1.6030	0.0500	0.0601	1.5370	0.0500	0.0679	1.5110	0.0500	0.0364	1.7640
	0.0250	0.0148	2.5920	0.0250	0.0266	2.0490	0.0250	0.0199	2.2830	0.0250	0.0295	1.9590
7	0.0100	0.0074	2.6720	0.0100	0.0088	2.3890	0.0100	0.0083	2.5190	0.0100	0.0077	2.4480
	0.0050	0.0074	2.6720	0.0050	0.0051	2.5620	0.0050	0.0083	2.5190	0.0050	0.0064	2.4490
	0.0025	0.0003	3.7410	0.0025	0.0012	3.0730	0.0025	0.0007	3.4290	0.0025	0.0020	2.8240
	0.0010	0.0003	3.7410	0.0010	0.0012	3.0730	0.0010	0.0007	3.4290	0.0010	0.0014	2.9390
				0.1000	0.0662	1.7880	0.1000	0.0984	1.3850	0.1000	0.0748	1.5630
				0.0500	0.0604	1.9400	0.0500	0.0447	1.7120	0.0500	0.0540	1.8440
				0.0250	0.0306	2.0000	0.0250	0.0335	1.9310	0.0250	0.0319	1.8970
8				0.0100	0.0050	2.9100	0.0100	0.0107	2.4140	0.0100	0.0070	2.5740
				0.0050	0.0050	2.9100	0.0050	0.0031	2.7710	0.0050	0.0055	2.7660
				0.0025	0.0025	3.0000	0.0025	0.0025	2.8970	0.0025	0.0029	2.8460
				0.0010	0.0000	4.0000	0.0010	0.0004	3.3800	0.0010	0.0002	3.6880
							0.1000	0.0810	1.4140	0.1000	0.0981	1.3690
							0.0500	0.0274	2.1080	0.0500	0.0502	1.7400
							0.0250	0.0251	2.2860	0.0250	0.0194	2.0220
9							0.0100	0.0125	2.3570	0.0100	0.0143	2.2830
							0.0050	0.0015	3.2010	0.0050	0.0045	2.6110
							0.0025	0.0015	3.2010	0.0025	0.0024	2.7400
							0.0010	0.0008	3.2990	0.0010	0.0010	3.0560
										0.1000	0.0894	1.4310
										0.0500	0.0398	1.7880
										0.0250	0.0117	2.4000
10										0.0100	0.0107	2.6030
										0.0050	0.0051	2.6830
										0.0025	0.0005	3.4700
										0.0010	0.0005	3.4700

Table C.2: Critical points, $\hat{T}_\alpha\{3\ldots10, 7\ldots10\}$, for the $\hat{T}$ distribution.

m, n	11			12			13			14		
3	0.1000	0.0947	1.7320	0.1000	0.1094	1.2450	0.1000	0.1015	1.7320	0.1000	0.0909	1.2450
	0.0500	0.0587	1.8610	0.0500	0.0684	1.7970	0.0500	0.0509	1.8410	0.0500	0.0410	1.8220
	0.0250	0.0105	1.9230	0.0250	0.0284	1.8680	0.0250	0.0076	1.8940	0.0250	0.0410	1.8220
	0.0100	0.0105	1.9230	0.0100	0.0021	1.9250	0.0100	0.0076	1.8940	0.0100	0.0060	1.8680
	0.0050	0.0105	1.9230	0.0050	0.0021	1.9250	0.0050	0.0076	1.8940	0.0050	0.0060	1.8680
	0.0025	0.0105	1.9230	0.0025	0.0021	1.9250	0.0025	0.0076	1.8940	0.0025	0.0060	1.8680
	0.0010	0.0105	1.9230	0.0010	0.0021	1.9250	0.0010	0.0076	1.8940	0.0010	0.0060	1.8680
4	0.1000	0.0832	1.1570	0.1000	0.1075	1.1330	0.1000	0.1072	1.1110	0.1000	0.0921	1.1150
	0.0500	0.0510	1.5820	0.0500	0.0463	1.1540	0.0500	0.0515	1.6110	0.0500	0.0451	1.1330
	0.0250	0.0249	2.1100	0.0250	0.0284	2.1820	0.0250	0.0222	2.1490	0.0250	0.0255	2.1560
	0.0100	0.0161	2.2270	0.0100	0.0095	2.2670	0.0100	0.0072	2.2230	0.0100	0.0067	2.2300
	0.0050	0.0029	2.3190	0.0050	0.0006	2.3090	0.0050	0.0072	2.2230	0.0050	0.0067	2.2300
	0.0025	0.0029	2.3190	0.0025	0.0006	2.3090	0.0025	0.0003	2.2780	0.0025	0.0003	2.2670
	0.0010	0.0029	2.3190	0.0010	0.0006	2.3090	0.0010	0.0003	2.2780	0.0010	0.0003	2.2670
5	0.1000	0.0781	1.5670	0.1000	0.0994	1.4760	0.1000	0.1183	1.4820	0.1000	0.0980	1.4570
	0.0500	0.0285	1.6180	0.0500	0.0657	1.5420	0.0500	0.0644	1.5470	0.0500	0.0587	1.5150
	0.0250	0.0285	1.6180	0.0250	0.0208	1.8750	0.0250	0.0239	1.5780	0.0250	0.0223	1.8630
	0.0100	0.0117	2.5120	0.0100	0.0093	2.3440	0.0100	0.0117	2.4700	0.0100	0.0096	2.4290
	0.0050	0.0064	2.6380	0.0050	0.0048	2.5710	0.0050	0.0047	2.5790	0.0050	0.0042	2.5250
	0.0025	0.0011	2.6960	0.0025	0.0007	2.6490	0.0025	0.0005	2.6310	0.0025	0.0042	2.5250
	0.0010	0.0011	2.6960	0.0010	0.0007	2.6490	0.0010	0.0005	2.6310	0.0010	0.0004	2.5930
6	0.1000	0.0876	1.3020	0.1000	0.1075	1.0000	0.1000	0.0914	1.2900	0.1000	0.0982	0.9750
	0.0500	0.0429	1.7360	0.0500	0.0599	1.8540	0.0500	0.0500	1.6320	0.0500	0.0585	1.8230
	0.0250	0.0172	2.0170	0.0250	0.0148	1.9990	0.0250	0.0312	1.9030	0.0250	0.0328	1.9100
	0.0100	0.0067	2.3080	0.0100	0.0148	1.9990	0.0100	0.0076	2.1510	0.0100	0.0121	1.9510
	0.0050	0.0044	2.5220	0.0050	0.0047	2.6110	0.0050	0.0047	2.4530	0.0050	0.0053	2.5880
	0.0025	0.0022	2.7700	0.0025	0.0030	2.9310	0.0025	0.0027	2.7210	0.0025	0.0024	2.8660
	0.0010	0.0004	3.0290	0.0010	0.0007	3.0000	0.0010	0.0003	2.9460	0.0010	0.0003	2.9270
7	0.1000	0.1164	1.4130	0.1000	0.1072	1.3680	0.1000	0.1114	1.3730	0.1000	0.1019	1.3360
	0.0500	0.0628	1.4500	0.0500	0.0412	1.6140	0.0500	0.0576	1.4060	0.0500	0.0427	1.6000
	0.0250	0.0250	2.0480	0.0250	0.0270	1.8920	0.0250	0.0262	2.1650	0.0250	0.0278	1.8420
	0.0100	0.0078	2.4170	0.0100	0.0075	2.3630	0.0100	0.0071	2.3440	0.0100	0.0067	2.3000
	0.0050	0.0078	2.4170	0.0050	0.0066	2.3650	0.0050	0.0071	2.3440	0.0050	0.0060	2.3020
	0.0025	0.0013	3.0990	0.0025	0.0025	2.7360	0.0025	0.0014	2.8330	0.0025	0.0025	2.6720
	0.0010	0.0011	3.2970	0.0010	0.0008	3.0150	0.0010	0.0009	3.2030	0.0010	0.0010	2.8000
8	0.1000	0.1062	1.3300	0.1000	0.0838	1.4140	0.1000	0.0949	1.3390	0.1000	0.0930	1.4140
	0.0500	0.0488	1.7550	0.0500	0.0584	1.7780	0.0500	0.0515	1.7210	0.0500	0.0532	1.7440
	0.0250	0.0348	1.8470	0.0250	0.0313	1.8250	0.0250	0.0302	1.7870	0.0250	0.0254	1.7720
	0.0100	0.0094	2.3090	0.0100	0.0100	2.3070	0.0100	0.0107	2.2340	0.0100	0.0106	2.5120
	0.0050	0.0051	2.4890	0.0050	0.0034	2.7380	0.0050	0.0057	2.5820	0.0050	0.0068	2.6160
	0.0025	0.0023	2.7710	0.0025	0.0034	2.7380	0.0025	0.0026	2.6810	0.0025	0.0024	2.6590
	0.0010	0.0005	3.1040	0.0010	0.0004	3.3350	0.0010	0.0008	2.8390	0.0010	0.0006	3.1760
9	0.1000	0.0847	1.3480	0.1000	0.0980	1.3140	0.1000	0.0787	1.3000	0.1000	0.0975	1.2730
	0.0500	0.0352	1.8460	0.0500	0.0511	1.6820	0.0500	0.0399	1.7870	0.0500	0.0474	1.6660
	0.0250	0.0254	2.1830	0.0250	0.0239	1.8120	0.0250	0.0259	2.1300	0.0250	0.0244	2.0150
	0.0100	0.0144	2.2470	0.0100	0.0056	2.3570	0.0100	0.0129	2.1680	0.0100	0.0069	2.1490
	0.0050	0.0028	2.8430	0.0050	0.0053	2.5230	0.0050	0.0038	2.6790	0.0050	0.0050	2.5470
	0.0025	0.0025	3.0480	0.0025	0.0033	2.6300	0.0025	0.0027	2.9820	0.0025	0.0024	2.6920
	0.0010	0.0013	3.1460	0.0010	0.0011	3.0670	0.0010	0.0012	3.0350	0.0010	0.0010	2.9720
10	0.1000	0.1096	1.3020	0.1000	0.0988	1.3740	0.1000	0.1146	1.2540	0.1000	0.0999	1.5250
	0.0500	0.0476	1.7360	0.0500	0.0424	1.7120	0.0500	0.0502	1.6720	0.0500	0.0411	1.6560
	0.0250	0.0230	1.8970	0.0250	0.0158	2.2350	0.0250	0.0253	1.9870	0.0250	0.0178	2.0380
	0.0100	0.0077	2.3060	0.0100	0.0109	2.5210	0.0100	0.0099	2.2440	0.0100	0.0115	2.4400
	0.0050	0.0056	2.6050	0.0050	0.0059	2.5690	0.0050	0.0062	2.5080	0.0050	0.0056	2.4840
	0.0025	0.0017	2.8970	0.0025	0.0010	3.1970	0.0025	0.0022	2.6190	0.0025	0.0014	2.9170
	0.0010	0.0008	3.0400	0.0010	0.0010	3.1970	0.0010	0.0013	2.9270	0.0010	0.0010	3.2530
11	0.1000	0.0920	1.2790	0.1000	0.1163	1.2440	0.1000	0.0977	1.2290	0.1000	0.1165	1.2030
	0.0500	0.0415	1.8280	0.0500	0.0551	1.6590	0.0500	0.0497	1.6400	0.0500	0.0480	1.6050
	0.0250	0.0176	2.1320	0.0250	0.0264	1.9830	0.0250	0.0204	2.0480	0.0250	0.0228	2.0060
	0.0100	0.0043	2.7710	0.0100	0.0096	2.1770	0.0100	0.0065	2.4920	0.0100	0.0101	2.3380
	0.0050	0.0043	2.7710	0.0050	0.0027	2.5680	0.0050	0.0053	2.8100	0.0050	0.0039	2.4320
	0.0025	0.0017	2.9840	0.0025	0.0025	2.7760	0.0025	0.0024	2.8670	0.0025	0.0026	2.8080
	0.0010	0.0017	2.9840	0.0010	0.0014	2.9040	0.0010	0.0003	3.4390	0.0010	0.0009	2.9630
12				0.1000	0.0925	1.6010	0.1000	0.0810	1.2550	0.1000	0.0964	1.5420
				0.0500	0.0489	1.6320	0.0500	0.0604	1.5940	0.0500	0.0524	1.5730
				0.0250	0.0196	2.1000	0.0250	0.0260	1.9930	0.0250	0.0239	1.8870
				0.0100	0.0080	2.4490	0.0100	0.0095	2.3920	0.0100	0.0095	2.3600
				0.0050	0.0080	2.4490	0.0050	0.0038	2.5430	0.0050	0.0025	2.7320
				0.0025	0.0017	3.0320	0.0025	0.0028	2.7900	0.0025	0.0025	2.7320
				0.0010	0.0008	3.2650	0.0010	0.0010	3.0870	0.0010	0.0009	3.1470
13							0.1000	0.1079	1.1760	0.1000	0.0873	1.1770
							0.0500	0.0483	1.9230	0.0500	0.0413	1.6350
							0.0250	0.0249	1.9610	0.0250	0.0303	1.9170
							0.0100	0.0088	2.3540	0.0100	0.0121	2.3010
							0.0050	0.0035	2.7450	0.0050	0.0050	2.6030
							0.0025	0.0035	2.7450	0.0025	0.0015	2.7970
							0.0010	0.0005	3.2770	0.0010	0.0011	3.0680
14										0.1000	0.1056	1.4820
										0.0500	0.0595	1.5110
										0.0250	0.0233	2.2230
										0.0100	0.0114	2.2670
										0.0050	0.0037	2.5920
										0.0025	0.0031	2.9640
										0.0010	0.0014	3.0230

Table C.3: Critical points, $\hat{T}_\alpha\{3\ldots14, 11\ldots14\}$, for the $\hat{T}$ distribution.

m, n	15			16			17			18		
	0.1000	0.1024	1.7320	0.1000	0.0925	1.2340	0.1000	0.1058	1.7320	0.1000	0.0936	1.2250
	0.0500	0.0410	1.8270	0.0500	0.0353	1.8110	0.0500	0.0350	1.8160	0.0500	0.0289	1.8020
	0.0250	0.0410	1.8270	0.0250	0.0353	1.8110	0.0250	0.0350	1.8160	0.0250	0.0289	1.8020
3	0.0100	0.0048	1.8730	0.0100	0.0040	1.8510	0.0100	0.0033	1.8570	0.0100	0.0029	1.8380
	0.0050	0.0048	1.8730	0.0050	0.0040	1.8510	0.0050	0.0033	1.8570	0.0050	0.0029	1.8380
	0.0025	0.0048	1.8730	0.0025	0.0040	1.8510	0.0025	0.0033	1.8570	0.0025	0.0029	1.8380
	0.0010	0.0048	1.8730	0.0010	0.0040	1.8510	0.0010	0.0033	1.8570	0.0010	0.0029	1.8380
	0.1000	0.0946	1.0970	0.1000	0.0810	1.1010	0.1000	0.0868	1.0860	0.1000	0.1015	1.0730
	0.0500	0.0511	1.5960	0.0500	0.0462	1.1180	0.0500	0.0525	1.5850	0.0500	0.0519	1.0950
	0.0250	0.0207	2.1290	0.0250	0.0231	2.1370	0.0250	0.0199	2.1140	0.0250	0.0299	2.0960
4	0.0100	0.0053	2.1940	0.0100	0.0051	2.2030	0.0100	0.0042	2.1720	0.0100	0.0086	2.1470
	0.0050	0.0053	2.1940	0.0050	0.0051	2.2030	0.0050	0.0042	2.1720	0.0050	0.0086	2.1470
	0.0025	0.0003	2.2420	0.0025	0.0002	2.2360	0.0025	0.0042	2.1720	0.0025	0.0007	2.1900
	0.0010	0.0003	2.2420	0.0010	0.0002	2.2360	0.0010	0.0001	2.2150	0.0010	0.0007	2.1900
	0.1000	0.1117	1.4630	0.1000	0.0945	1.4430	0.1000	0.0741	1.4710	0.1000	0.0897	1.4310
	0.0500	0.0559	1.5210	0.0500	0.0525	1.4940	0.0500	0.0317	1.5090	0.0500	0.0481	1.4770
	0.0250	0.0229	1.5490	0.0250	0.0232	1.8540	0.0250	0.0189	1.5260	0.0250	0.0265	1.5100
5	0.0100	0.0121	2.4390	0.0100	0.0093	2.4050	0.0100	0.0072	2.4520	0.0100	0.0089	2.3860
	0.0050	0.0043	2.5350	0.0050	0.0034	2.4900	0.0050	0.0072	2.4520	0.0050	0.0028	2.4620
	0.0025	0.0043	2.5350	0.0025	0.0034	2.4900	0.0025	0.0013	2.5150	0.0025	0.0028	2.4620
	0.0010	0.0004	2.5810	0.0010	0.0003	2.5510	0.0010	0.0013	2.5150	0.0010	0.0003	2.5180
	0.1000	0.0925	1.2810	0.1000	0.0842	0.9570	0.1000	0.0953	1.2750	0.1000	0.0833	0.9420
	0.0500	0.0465	1.7900	0.0500	0.0464	1.8330	0.0500	0.0474	1.7720	0.0500	0.0408	1.8110
	0.0250	0.0288	1.8680	0.0250	0.0210	1.8790	0.0250	0.0277	1.8410	0.0250	0.0171	1.8510
6	0.0100	0.0086	2.1360	0.0100	0.0080	1.9140	0.0100	0.0090	2.1250	0.0100	0.0078	1.8850
	0.0050	0.0051	2.4040	0.0050	0.0055	2.6480	0.0050	0.0048	2.4490	0.0050	0.0057	2.6260
	0.0025	0.0018	2.8020	0.0025	0.0036	2.7500	0.0025	0.0015	2.7620	0.0025	0.0034	2.7170
	0.0010	0.0003	2.8850	0.0010	0.0010	2.8340	0.0010	0.0015	2.7620	0.0010	0.0007	2.7940
	0.1000	0.0872	1.3460	0.1000	0.0968	1.2820	0.1000	0.0791	1.3210	0.1000	0.1155	1.2650
	0.0500	0.0453	1.3730	0.0500	0.0448	1.5890	0.0500	0.0444	1.3470	0.0500	0.0531	1.3320
	0.0250	0.0242	2.1760	0.0250	0.0236	1.8890	0.0250	0.0232	2.1430	0.0250	0.0246	2.0330
7	0.0100	0.0128	2.2540	0.0100	0.0132	2.1860	0.0100	0.0110	2.2140	0.0100	0.0099	2.1750
	0.0050	0.0045	2.2880	0.0050	0.0056	2.2550	0.0050	0.0040	2.2450	0.0050	0.0045	2.2190
	0.0025	0.0020	2.7740	0.0025	0.0026	2.6240	0.0025	0.0022	2.7590	0.0025	0.0025	2.6100
	0.0010	0.0006	3.1560	0.0010	0.0011	2.9240	0.0010	0.0005	3.1000	0.0010	0.0011	2.9510
	0.1000	0.0975	1.3050	0.1000	0.0959	1.4140	0.1000	0.1070	1.2530	0.1000	0.1007	1.4790
	0.0500	0.0528	1.6820	0.0500	0.0494	1.6970	0.0500	0.0443	1.6500	0.0500	0.0456	1.6660
	0.0250	0.0191	1.8770	0.0250	0.0232	1.7320	0.0250	0.0264	1.7090	0.0250	0.0227	1.6990
8	0.0100	0.0086	2.2530	0.0100	0.0109	2.4650	0.0100	0.0099	2.2150	0.0100	0.0109	2.4280
	0.0050	0.0058	2.5240	0.0050	0.0062	2.5580	0.0050	0.0049	2.5060	0.0050	0.0054	2.5120
	0.0025	0.0028	2.6130	0.0025	0.0020	2.5980	0.0025	0.0022	2.5620	0.0025	0.0019	2.5490
	0.0010	0.0011	2.9440	0.0010	0.0006	2.9760	0.0010	0.0010	2.9230	0.0010	0.0008	2.9590
	0.1000	0.0751	1.2640	0.1000	0.0853	1.2450	0.1000	0.1188	1.2170	0.1000	0.1151	1.1920
	0.0500	0.0456	1.7710	0.0500	0.0544	1.6190	0.0500	0.0491	1.6660	0.0500	0.0466	1.6260
	0.0250	0.0263	2.0730	0.0250	0.0226	2.0040	0.0250	0.0254	2.0280	0.0250	0.0220	1.9640
9	0.0100	0.0121	2.1080	0.0100	0.0077	2.2190	0.0100	0.0117	2.0610	0.0100	0.0082	2.0940
	0.0050	0.0044	2.4790	0.0050	0.0053	2.4900	0.0050	0.0053	2.5990	0.0050	0.0052	2.4390
	0.0025	0.0025	2.9020	0.0025	0.0024	2.7190	0.0025	0.0027	2.8400	0.0025	0.0019	2.7810
	0.0010	0.0009	2.9510	0.0010	0.0007	2.9050	0.0010	0.0010	2.8850	0.0010	0.0009	2.8460
	0.1000	0.1047	1.2200	0.1000	0.0999	1.5180	0.1000	0.1062	1.1910	0.1000	0.1008	1.4910
	0.0500	0.0493	1.6260	0.0500	0.0393	1.6120	0.0500	0.0469	1.5870	0.0500	0.0382	1.5770
	0.0250	0.0270	1.9780	0.0250	0.0207	2.0210	0.0250	0.0226	1.9840	0.0250	0.0232	1.8970
10	0.0100	0.0102	2.3740	0.0100	0.0123	2.3770	0.0100	0.0105	2.3210	0.0100	0.0121	2.3280
	0.0050	0.0056	2.4400	0.0050	0.0060	2.4180	0.0050	0.0057	2.3820	0.0050	0.0056	2.3660
	0.0025	0.0026	2.7700	0.0025	0.0018	2.8690	0.0025	0.0027	2.7080	0.0025	0.0022	2.6760
	0.0010	0.0010	3.0230	0.0010	0.0010	3.1700	0.0010	0.0010	2.9670	0.0010	0.0012	3.1040
	0.1000	0.0983	1.1900	0.1000	0.1145	1.1700	0.1000	0.0957	1.1600	0.1000	0.1116	1.1430
	0.0500	0.0517	1.8600	0.0500	0.0414	1.6050	0.0500	0.0539	1.8200	0.0500	0.0453	1.5940
	0.0250	0.0208	1.9840	0.0250	0.0233	1.9500	0.0250	0.0201	1.9340	0.0250	0.0237	1.9050
11	0.0100	0.0078	2.2720	0.0100	0.0092	2.3400	0.0100	0.0090	2.2530	0.0100	0.0104	2.2860
	0.0050	0.0053	2.7290	0.0050	0.0048	2.3920	0.0050	0.0053	2.6620	0.0050	0.0047	2.5770
	0.0025	0.0027	2.7780	0.0025	0.0026	2.7300	0.0025	0.0025	2.7080	0.0025	0.0024	2.6690
	0.0010	0.0004	3.1390	0.0010	0.0010	3.0360	0.0010	0.0007	3.0900	0.0010	0.0011	2.9710
	0.1000	0.0910	1.2420	0.1000	0.0957	1.4960	0.1000	0.0990	1.1540	0.1000	0.0941	1.4600
	0.0500	0.0538	1.5430	0.0500	0.0536	1.5270	0.0500	0.0568	1.5020	0.0500	0.0533	1.4900
	0.0250	0.0300	1.9280	0.0250	0.0252	2.1450	0.0250	0.0196	1.9690	0.0250	0.0270	2.1010
12	0.0100	0.0112	2.3140	0.0100	0.0100	2.2910	0.0100	0.0097	2.2530	0.0100	0.0100	2.2360
	0.0050	0.0049	2.4480	0.0050	0.0031	2.6800	0.0050	0.0051	2.5560	0.0050	0.0038	2.4700
	0.0025	0.0029	2.7000	0.0025	0.0024	2.9930	0.0025	0.0021	2.7850	0.0025	0.0025	2.9290
	0.0010	0.0009	3.0860	0.0010	0.0010	3.0550	0.0010	0.0009	3.0040	0.0010	0.0012	2.9810
	0.1000	0.1124	1.1360	0.1000	0.0926	1.1940	0.1000	0.1125	1.1050	0.1000	0.0954	1.1850
	0.0500	0.0511	1.8570	0.0500	0.0475	1.4930	0.0500	0.0464	1.8080	0.0500	0.0513	1.6790
	0.0250	0.0263	1.8940	0.0250	0.0321	1.8590	0.0250	0.0283	1.8420	0.0250	0.0202	1.8970
13	0.0100	0.0099	2.4530	0.0100	0.0073	2.2470	0.0100	0.0098	2.5270	0.0100	0.0092	2.2130
	0.0050	0.0037	2.6520	0.0050	0.0047	2.5680	0.0050	0.0049	2.5790	0.0050	0.0051	2.5370
	0.0025	0.0037	2.6520	0.0025	0.0019	2.6980	0.0025	0.0013	2.6950	0.0025	0.0025	2.6200
	0.0010	0.0008	3.1540	0.0010	0.0010	2.9750	0.0010	0.0010	3.2490	0.0010	0.0009	3.0220
	0.1000	0.0968	1.1630	0.1000	0.0908	1.4360	0.1000	0.1025	1.3240	0.1000	0.1056	1.4050
	0.0500	0.0452	1.5570	0.0500	0.0634	1.4630	0.0500	0.0494	1.4400	0.0500	0.0622	1.4250
	0.0250	0.0194	1.8690	0.0250	0.0271	2.1520	0.0250	0.0236	1.8590	0.0250	0.0269	2.1080
14	0.0100	0.0064	2.3610	0.0100	0.0135	2.1950	0.0100	0.0080	2.1690	0.0100	0.0137	2.1380
	0.0050	0.0049	2.5240	0.0050	0.0045	2.3330	0.0050	0.0046	2.5160	0.0050	0.0053	2.7170
	0.0025	0.0020	2.7130	0.0025	0.0029	2.8720	0.0025	0.0024	2.7910	0.0025	0.0017	2.8500
	0.0010	0.0014	2.9600	0.0010	0.0004	3.3860	0.0010	0.0008	2.9350	0.0010	0.0005	3.3960

Table C.4: Critical points, $\hat{T}_\alpha\{3\dots14, 15\dots18\}$, for the $\hat{T}$ distribution.

m, n	**15**			**16**			**17**			**18**		
15	0.1000	0.1200	1.0950	0.1000	0.1011	1.3050	0.1000	0.0787	1.3310	0.1000	0.1027	1.3470
	0.0500	0.0590	1.7900	0.0500	0.0504	1.6400	0.0500	0.0508	1.7470	0.0500	0.0515	1.6890
	0.0250	0.0318	1.8250	0.0250	0.0224	1.8080	0.0250	0.0165	2.0890	0.0250	0.0252	1.9230
	0.0100	0.0111	2.5060	0.0100	0.0089	2.2840	0.0100	0.0106	2.4460	0.0100	0.0098	2.3570
	0.0050	0.0053	2.5560	0.0050	0.0067	2.5050	0.0050	0.0066	2.4790	0.0050	0.0035	2.5290
	0.0025	0.0013	2.8180	0.0025	0.0024	2.7750	0.0025	0.0024	2.8690	0.0025	0.0025	2.7850
	0.0010	0.0011	3.2220	0.0010	0.0008	2.9520	0.0010	0.0008	3.1880	0.0010	0.0010	2.8850
16				0.1000	0.1167	1.3940	0.1000	0.0987	1.3440	0.1000	0.1014	1.3560
				0.0500	0.0381	1.7650	0.0500	0.0479	1.6810	0.0500	0.0429	1.6150
				0.0250	0.0301	2.0920	0.0250	0.0255	2.0150	0.0250	0.0268	2.0340
				0.0100	0.0060	2.5290	0.0100	0.0105	2.3510	0.0100	0.0078	2.3140
				0.0050	0.0047	2.7890	0.0050	0.0043	2.5300	0.0050	0.0049	2.7120
				0.0025	0.0023	2.8280	0.0025	0.0030	2.7750	0.0025	0.0029	2.7480
				0.0010	0.0004	3.1620	0.0010	0.0009	3.0260	0.0010	0.0008	3.0910
17							0.1000	0.0853	1.3170	0.1000	0.1002	1.3050
							0.0500	0.0380	1.7140	0.0500	0.0488	1.6840
							0.0250	0.0183	1.9970	0.0250	0.0244	1.9580
							0.0100	0.0072	2.4000	0.0100	0.0093	2.2940
							0.0050	0.0072	2.4000	0.0050	0.0046	2.3770
							0.0025	0.0026	2.9280	0.0025	0.0034	2.6940
							0.0010	0.0010	3.0860	0.0010	0.0011	3.0310
18										0.1000	0.0784	1.3330
										0.0500	0.0474	1.5360
										0.0250	0.0201	1.9990
										0.0100	0.0088	2.2180
										0.0050	0.0032	2.6660
										0.0025	0.0032	2.6660
										0.0010	0.0011	3.1620

Table C.5: Critical points, $\hat{T}_\alpha\{15\ldots18, 15\ldots18\}$, for the $\hat{T}$ distribution.

m, n	19			20			21			22		
3	0.1000	0.0954	1.7600	0.1000	0.0964	1.2180	0.1000	0.0938	1.7570	0.1000	0.0997	1.2020
	0.0500	0.0467	1.7910	0.0500	0.0729	1.7620	0.0500	0.0406	1.7850	0.0500	0.0345	1.7780
	0.0250	0.0063	1.8230	0.0250	0.0258	1.7950	0.0250	0.0406	1.7850	0.0250	0.0345	1.7780
	0.0100	0.0063	1.8230	0.0100	0.0022	1.8270	0.0100	0.0054	1.8140	0.0100	0.0044	1.8030
	0.0050	0.0063	1.8230	0.0050	0.0022	1.8270	0.0050	0.0054	1.8140	0.0050	0.0044	1.8030
	0.0025	0.0063	1.8230	0.0025	0.0022	1.8270	0.0025	0.0054	1.8140	0.0025	0.0044	1.8030
	0.0010	0.0063	1.8230	0.0010	0.0022	1.8270	0.0010	0.0054	1.8140	0.0010	0.0044	1.8030
4	0.1000	0.0803	1.0770	0.1000	0.0902	1.0660	0.1000	0.0945	1.0570	0.1000	0.0840	1.0600
	0.0500	0.0526	1.5760	0.0500	0.0511	1.0860	0.0500	0.0465	1.5860	0.0500	0.0516	1.0780
	0.0250	0.0186	2.1020	0.0250	0.0271	2.0860	0.0250	0.0228	2.0740	0.0250	0.0243	2.0790
	0.0100	0.0033	2.1540	0.0100	0.0074	2.1320	0.0100	0.0058	2.1140	0.0100	0.0055	2.1200
	0.0050	0.0033	2.1540	0.0050	0.0074	2.1320	0.0050	0.0058	2.1140	0.0050	0.0055	2.1200
	0.0025	0.0033	2.1540	0.0025	0.0006	2.1720	0.0025	0.0004	2.1510	0.0025	0.0003	2.1560
	0.0010	0.0001	2.1930	0.0010	0.0006	2.1720	0.0010	0.0004	2.1510	0.0010	0.0003	2.1560
5	0.1000	0.1205	1.4180	0.1000	0.1037	1.4070	0.1000	0.1118	1.4110	0.1000	0.0994	1.4010
	0.0500	0.0652	1.4580	0.0500	0.0606	1.4420	0.0500	0.0572	1.4470	0.0500	0.0555	1.4330
	0.0250	0.0289	1.4920	0.0250	0.0240	1.4960	0.0250	0.0268	1.4780	0.0250	0.0254	1.8270
	0.0100	0.0058	2.4300	0.0100	0.0111	2.3450	0.0100	0.0135	2.3510	0.0100	0.0110	2.3350
	0.0050	0.0058	2.4300	0.0050	0.0045	2.4030	0.0050	0.0048	2.4120	0.0050	0.0042	2.3880
	0.0025	0.0009	2.4860	0.0025	0.0006	2.4570	0.0025	0.0006	2.4630	0.0025	0.0042	2.3880
	0.0010	0.0009	2.4860	0.0010	0.0006	2.4570	0.0010	0.0006	2.4630	0.0010	0.0006	2.4370
6	0.1000	0.0992	0.9330	0.1000	0.1122	0.9200	0.1000	0.0988	1.2570	0.1000	0.1067	0.9110
	0.0500	0.0538	1.7340	0.0500	0.0383	1.7940	0.0500	0.0547	1.7240	0.0500	0.0628	1.7290
	0.0250	0.0179	1.8350	0.0250	0.0160	1.8290	0.0250	0.0171	1.8160	0.0250	0.0344	1.7790
	0.0100	0.0099	2.1020	0.0100	0.0083	1.8610	0.0100	0.0101	2.0960	0.0100	0.0083	1.8420
	0.0050	0.0050	2.5220	0.0050	0.0061	2.6080	0.0050	0.0048	2.5860	0.0050	0.0058	2.5940
	0.0025	0.0026	2.6800	0.0025	0.0028	2.6910	0.0025	0.0025	2.6590	0.0025	0.0025	2.6690
	0.0010	0.0007	2.7530	0.0010	0.0006	2.7610	0.0010	0.0005	2.7250	0.0010	0.0004	2.7330
7	0.1000	0.1211	1.2700	0.1000	0.1108	1.2520	0.1000	0.1143	1.2570	0.1000	0.1070	1.2410
	0.0500	0.0427	1.3260	0.0500	0.0524	1.3130	0.0500	0.0438	1.3090	0.0500	0.0500	1.5570
	0.0250	0.0220	2.1170	0.0250	0.0259	2.0190	0.0250	0.0204	2.0960	0.0250	0.0273	2.0070
	0.0100	0.0100	2.1820	0.0100	0.0089	2.1480	0.0100	0.0087	2.1420	0.0100	0.0092	2.1250
	0.0050	0.0042	2.2100	0.0050	0.0044	2.1880	0.0050	0.0038	2.1820	0.0050	0.0050	2.1630
	0.0025	0.0027	2.8560	0.0025	0.0026	2.5780	0.0025	0.0028	2.8360	0.0025	0.0024	2.5950
	0.0010	0.0007	3.0540	0.0010	0.0011	2.9210	0.0010	0.0016	2.9350	0.0010	0.0013	2.8970
8	0.1000	0.1049	1.2330	0.1000	0.1028	1.4730	0.1000	0.1058	1.2180	0.1000	0.1053	1.4680
	0.0500	0.0405	1.6230	0.0500	0.0402	1.6410	0.0500	0.0402	1.6010	0.0500	0.0371	1.6200
	0.0250	0.0251	1.6800	0.0250	0.0216	1.6730	0.0250	0.0264	1.6560	0.0250	0.0220	1.6510
	0.0100	0.0105	2.2960	0.0100	0.0103	2.3980	0.0100	0.0088	2.3610	0.0100	0.0106	2.3730
	0.0050	0.0046	2.4670	0.0050	0.0046	2.4620	0.0050	0.0047	2.4360	0.0050	0.0048	2.4310
	0.0025	0.0021	2.5190	0.0025	0.0018	2.5090	0.0025	0.0022	2.4830	0.0025	0.0016	2.4770
	0.0010	0.0010	2.8780	0.0010	0.0009	2.9460	0.0010	0.0011	2.8420	0.0010	0.0009	2.9360
9	0.1000	0.1109	1.1950	0.1000	0.1110	1.1730	0.1000	0.1053	1.1780	0.1000	0.1064	1.1580
	0.0500	0.0505	1.7490	0.0500	0.0480	1.5990	0.0500	0.0506	1.8210	0.0500	0.0501	1.5760
	0.0250	0.0236	1.9830	0.0250	0.0214	1.9310	0.0250	0.0223	1.9530	0.0250	0.0208	1.9050
	0.0100	0.0112	2.0230	0.0100	0.0090	2.0850	0.0100	0.0108	1.9920	0.0100	0.0102	2.1600
	0.0050	0.0049	2.6920	0.0050	0.0054	2.4000	0.0050	0.0051	2.6590	0.0050	0.0051	2.5200
	0.0025	0.0024	2.7760	0.0025	0.0019	2.7380	0.0025	0.0026	2.7490	0.0025	0.0022	2.7030
	0.0010	0.0009	2.8320	0.0010	0.0008	2.7800	0.0010	0.0009	2.7880	0.0010	0.0009	2.7590
10	0.1000	0.0924	1.2640	0.1000	0.0979	1.4200	0.1000	0.1014	1.3220	0.1000	0.1098	1.4210
	0.0500	0.0447	1.5560	0.0500	0.0385	1.5490	0.0500	0.0458	1.5300	0.0500	0.0517	1.5090
	0.0250	0.0273	1.8990	0.0250	0.0246	2.1010	0.0250	0.0249	1.8970	0.0250	0.0261	2.0510
	0.0100	0.0104	2.2790	0.0100	0.0115	2.2880	0.0100	0.0108	2.2450	0.0100	0.0092	2.2570
	0.0050	0.0055	2.3350	0.0050	0.0054	2.3230	0.0050	0.0058	2.2970	0.0050	0.0043	2.2880
	0.0025	0.0022	2.7240	0.0025	0.0027	2.8020	0.0025	0.0024	2.6440	0.0025	0.0025	2.8420
	0.0010	0.0010	3.0390	0.0010	0.0013	3.0510	0.0010	0.0008	2.9930	0.0010	0.0008	3.0100
11	0.1000	0.0928	1.1360	0.1000	0.1009	1.1220	0.1000	0.0848	1.1160	0.1000	0.1021	1.1040
	0.0500	0.0548	1.7880	0.0500	0.0496	1.5070	0.0500	0.0504	1.7880	0.0500	0.0518	1.6540
	0.0250	0.0204	1.8940	0.0250	0.0232	1.8690	0.0250	0.0312	1.8400	0.0250	0.0229	1.8390
	0.0100	0.0095	2.4350	0.0100	0.0099	2.2440	0.0100	0.0108	2.4090	0.0100	0.0102	2.2080
	0.0050	0.0063	2.6080	0.0050	0.0050	2.5540	0.0050	0.0050	2.5690	0.0050	0.0048	2.4870
	0.0025	0.0028	2.6520	0.0025	0.0025	2.6180	0.0025	0.0021	2.6050	0.0025	0.0025	2.5760
	0.0010	0.0009	2.8770	0.0010	0.0009	2.9910	0.0010	0.0011	3.0980	0.0010	0.0011	2.8750
12	0.1000	0.0984	1.3220	0.1000	0.1123	1.3990	0.1000	0.1013	1.3480	0.1000	0.1103	1.3780
	0.0500	0.0429	1.5290	0.0500	0.0496	1.4600	0.0500	0.0459	1.4530	0.0500	0.0487	1.4350
	0.0250	0.0226	1.9450	0.0250	0.0273	2.0990	0.0250	0.0251	1.8250	0.0250	0.0272	2.0670
	0.0100	0.0107	2.2040	0.0100	0.0091	2.1900	0.0100	0.0101	2.1630	0.0100	0.0090	2.1530
	0.0050	0.0050	2.5700	0.0050	0.0047	2.4290	0.0050	0.0047	2.5220	0.0050	0.0048	2.6500
	0.0025	0.0027	2.7370	0.0025	0.0026	2.8800	0.0025	0.0027	2.6970	0.0025	0.0021	2.8310
	0.0010	0.0010	2.9380	0.0010	0.0010	2.9210	0.0010	0.0011	2.8840	0.0010	0.0009	2.8700
13	0.1000	0.1072	1.0790	0.1000	0.0985	1.1570	0.1000	0.1070	1.0580	0.1000	0.0994	1.2520
	0.0500	0.0497	1.7740	0.0500	0.0536	1.6500	0.0500	0.0488	1.7390	0.0500	0.0472	1.6820
	0.0250	0.0272	1.7990	0.0250	0.0219	1.8740	0.0250	0.0271	1.7640	0.0250	0.0232	1.7570
	0.0100	0.0097	2.4840	0.0100	0.0100	2.1870	0.0100	0.0091	2.4350	0.0100	0.0104	2.2780
	0.0050	0.0044	2.5190	0.0050	0.0053	2.4830	0.0050	0.0042	2.4700	0.0050	0.0052	2.4380
	0.0025	0.0016	2.7990	0.0025	0.0026	2.7630	0.0025	0.0019	2.6290	0.0025	0.0026	2.7170
	0.0010	0.0010	3.1940	0.0010	0.0010	2.9710	0.0010	0.0010	3.1380	0.0010	0.0009	3.0290
14	0.1000	0.0987	1.3560	0.1000	0.0951	1.3760	0.1000	0.0980	1.3280	0.1000	0.1008	1.3480
	0.0500	0.0495	1.6950	0.0500	0.0401	1.7000	0.0500	0.0507	1.6600	0.0500	0.0427	1.6910
	0.0250	0.0262	1.9480	0.0250	0.0238	2.0640	0.0250	0.0237	1.9880	0.0250	0.0259	2.0220
	0.0100	0.0095	2.1380	0.0100	0.0064	2.4030	0.0100	0.0100	2.2360	0.0100	0.0074	2.2550
	0.0050	0.0039	2.5370	0.0050	0.0054	2.6620	0.0050	0.0045	2.4150	0.0050	0.0046	2.6960
	0.0025	0.0025	2.8060	0.0025	0.0019	2.7870	0.0025	0.0025	2.7480	0.0025	0.0021	2.7350
	0.0010	0.0010	2.9220	0.0010	0.0006	3.1750	0.0010	0.0011	3.0100	0.0010	0.0008	2.9750

Table C.6: Critical points, $\hat{T}_\alpha\{3\ldots14, 19\ldots22\}$, for the $\hat{T}$ distribution.

m, n	19			20			21			22		
	0.1000	0.0824	1.3790	0.1000	0.1008	1.3220	0.1000	0.0873	1.3700	0.1000	0.1018	1.3080
	0.0500	0.0531	1.7040	0.0500	0.0543	1.6520	0.0500	0.0537	1.6690	0.0500	0.0537	1.6360
	0.0250	0.0190	1.9300	0.0250	0.0255	1.9730	0.0250	0.0202	1.9190	0.0250	0.0261	1.9630
15	0.0100	0.0116	2.3860	0.0100	0.0110	2.3140	0.0100	0.0122	2.3360	0.0100	0.0112	2.2900
	0.0050	0.0067	2.4170	0.0050	0.0042	2.4620	0.0050	0.0069	2.3660	0.0050	0.0047	2.4150
	0.0025	0.0025	2.7490	0.0025	0.0031	2.7210	0.0025	0.0025	2.9040	0.0025	0.0029	2.6710
	0.0010	0.0008	3.1080	0.0010	0.0010	3.0610	0.0010	0.0007	3.0420	0.0010	0.0009	3.0050
	0.1000	0.1032	1.3110	0.1000	0.1024	1.3240	0.1000	0.1066	1.2950	0.1000	0.1065	1.2970
	0.0500	0.0516	1.6390	0.0500	0.0456	1.6030	0.0500	0.0494	1.6540	0.0500	0.0502	1.6930
	0.0250	0.0258	1.9670	0.0250	0.0282	1.9860	0.0250	0.0231	1.9850	0.0250	0.0307	1.9460
16	0.0100	0.0102	2.2950	0.0100	0.0085	2.2830	0.0100	0.0108	2.3160	0.0100	0.0101	2.3900
	0.0050	0.0047	2.4550	0.0050	0.0049	2.6480	0.0050	0.0055	2.5910	0.0050	0.0061	2.5950
	0.0025	0.0031	2.7030	0.0025	0.0028	2.6830	0.0025	0.0020	2.6740	0.0025	0.0034	2.6280
	0.0010	0.0010	3.0410	0.0010	0.0009	2.8540	0.0010	0.0010	2.9780	0.0010	0.0009	3.2390
	0.1000	0.0908	1.2570	0.1000	0.0964	1.3150	0.1000	0.0965	1.2120	0.1000	0.0998	1.2870
	0.0500	0.0586	1.6470	0.0500	0.0505	1.6440	0.0500	0.0428	1.6310	0.0500	0.0505	1.6090
	0.0250	0.0224	1.8290	0.0250	0.0257	1.9730	0.0250	0.0250	1.9410	0.0250	0.0259	1.9310
17	0.0100	0.0091	2.3360	0.0100	0.0115	2.3020	0.0100	0.0094	2.2830	0.0100	0.0095	2.2540
	0.0050	0.0037	2.5280	0.0050	0.0049	2.6310	0.0050	0.0041	2.4960	0.0050	0.0048	2.5750
	0.0025	0.0028	2.9630	0.0025	0.0026	2.7820	0.0025	0.0023	2.8980	0.0025	0.0023	2.8280
	0.0010	0.0012	3.0040	0.0010	0.0007	3.0670	0.0010	0.0014	2.9360	0.0010	0.0009	2.9960
	0.1000	0.0962	1.3110	0.1000	0.1110	1.2820	0.1000	0.0995	1.2810	0.1000	0.1148	1.2540
	0.0500	0.0525	1.6390	0.0500	0.0498	1.8460	0.0500	0.0563	1.6010	0.0500	0.0492	1.8150
	0.0250	0.0254	1.9670	0.0250	0.0227	1.9490	0.0250	0.0272	1.9210	0.0250	0.0238	1.9060
18	0.0100	0.0112	2.2950	0.0100	0.0100	2.4700	0.0100	0.0076	2.2840	0.0100	0.0108	2.4200
	0.0050	0.0054	2.5640	0.0050	0.0043	2.5990	0.0050	0.0050	2.5620	0.0050	0.0042	2.5420
	0.0025	0.0022	2.7590	0.0025	0.0016	2.8900	0.0025	0.0025	2.8190	0.0025	0.0016	2.6980
	0.0010	0.0008	3.0650	0.0010	0.0011	3.2050	0.0010	0.0009	2.9870	0.0010	0.0010	3.1360
	0.1000	0.0985	1.2460	0.1000	0.1040	1.2770	0.1000	0.1009	1.4230	0.1000	0.1085	1.2490
	0.0500	0.0459	1.6220	0.0500	0.0564	1.5970	0.0500	0.0484	1.5830	0.0500	0.0589	1.5610
	0.0250	0.0253	2.0310	0.0250	0.0289	1.9160	0.0250	0.0250	2.1060	0.0250	0.0202	1.9130
19	0.0100	0.0102	2.2710	0.0100	0.0074	2.2490	0.0100	0.0113	2.2160	0.0100	0.0084	2.1860
	0.0050	0.0043	2.4290	0.0050	0.0053	2.5550	0.0050	0.0050	2.5610	0.0050	0.0063	2.4980
	0.0025	0.0033	2.8800	0.0025	0.0024	2.8100	0.0025	0.0021	2.8490	0.0025	0.0022	2.8100
	0.0010	0.0005	3.1920	0.0010	0.0009	2.9860	0.0010	0.0007	3.1310	0.0010	0.0010	3.0490
				0.1000	0.0876	1.2640	0.1000	0.1071	1.2460	0.1000	0.0914	1.2350
				0.0500	0.0537	1.8000	0.0500	0.0418	1.6450	0.0500	0.0472	1.8280
				0.0250	0.0249	1.8970	0.0250	0.0201	1.9200	0.0250	0.0271	1.8530
20				0.0100	0.0106	2.4950	0.0100	0.0091	2.1940	0.0100	0.0089	2.4380
				0.0050	0.0052	2.5290	0.0050	0.0062	2.4920	0.0050	0.0058	2.4710
				0.0025	0.0021	2.6310	0.0025	0.0024	2.7590	0.0025	0.0025	2.9350
				0.0010	0.0007	3.1620	0.0010	0.0010	3.0450	0.0010	0.0011	3.0890
							0.1000	0.1013	1.4630	0.1000	0.1132	1.2160
							0.0500	0.0528	1.5430	0.0500	0.0452	1.6070
							0.0250	0.0253	2.1300	0.0250	0.0234	1.8750
21							0.0100	0.0063	2.3100	0.0100	0.0103	2.2750
							0.0050	0.0049	2.7390	0.0050	0.0046	2.5590
							0.0025	0.0024	2.7770	0.0025	0.0031	2.7370
							0.0010	0.0009	3.0360	0.0010	0.0010	3.0410
										0.1000	0.0949	1.2060
										0.0500	0.0498	1.7900
										0.0250	0.0295	1.8090
22										0.0100	0.0132	2.3870
										0.0050	0.0067	2.4120
										0.0025	0.0026	2.9840
										0.0010	0.0012	3.0150

Table C.7: Critical points, $\hat{T}_\alpha\{15\ldots22, 19\ldots22\}$, for the $\hat{T}$ distribution.

m, n	23			24			25		
	0.1000	0.1106	1.7320	0.1000	0.0999	1.1980	0.1000	0.1107	1.7320
	0.0500	0.0344	1.7810	0.0500	0.0301	1.7740	0.0500	0.0303	1.7770
	0.0250	0.0344	1.7810	0.0250	0.0301	1.7740	0.0250	0.0303	1.7770
3	0.0100	0.0039	1.8070	0.0100	0.0032	1.7970	0.0100	0.0033	1.8010
	0.0050	0.0039	1.8070	0.0050	0.0032	1.7970	0.0050	0.0033	1.8010
	0.0025	0.0039	1.8070	0.0025	0.0032	1.7970	0.0025	0.0033	1.8010
	0.0010	0.0039	1.8070	0.0010	0.0032	1.7970	0.0010	0.0033	1.8010
	0.1000	0.0861	1.0520	0.1000	0.0791	1.0550	0.1000	0.0809	1.0480
	0.0500	0.0538	1.5510	0.0500	0.0499	2.0000	0.0500	0.0469	1.5720
	0.0250	0.0212	2.0680	0.0250	0.0229	2.0720	0.0250	0.0194	2.0620
4	0.0100	0.0047	2.1040	0.0100	0.0047	2.1100	0.0100	0.0041	2.0960
	0.0050	0.0047	2.1040	0.0050	0.0047	2.1100	0.0050	0.0041	2.0960
	0.0025	0.0003	2.1380	0.0025	0.0003	2.1440	0.0025	0.0041	2.0960
	0.0010	0.0003	2.1380	0.0010	0.0003	2.1440	0.0010	0.0003	2.1270
	0.1000	0.1064	1.4050	0.1000	0.0958	1.3960	0.1000	0.1013	1.4000
	0.0500	0.0519	1.4380	0.0500	0.0512	1.4250	0.0500	0.0489	1.4300
	0.0250	0.0261	1.4660	0.0250	0.0249	1.4710	0.0250	0.0253	1.4570
5	0.0100	0.0132	2.3410	0.0100	0.0101	2.3270	0.0100	0.0122	2.3330
	0.0050	0.0044	2.3970	0.0050	0.0033	2.3760	0.0050	0.0033	2.3840
	0.0025	0.0044	2.3970	0.0025	0.0033	2.3760	0.0025	0.0033	2.3840
	0.0010	0.0005	2.4440	0.0010	0.0004	2.4210	0.0010	0.0004	2.4280
	0.1000	0.1001	0.9140	0.1000	0.1036	0.9030	0.1000	0.1005	1.2520
	0.0500	0.0538	1.7160	0.0500	0.0609	1.7210	0.0500	0.0526	1.7090
	0.0250	0.0313	1.7600	0.0250	0.0318	1.7670	0.0250	0.0289	1.7500
6	0.0100	0.0104	2.1450	0.0100	0.0088	1.8250	0.0100	0.0101	2.1370
	0.0050	0.0048	2.5740	0.0050	0.0058	2.5820	0.0050	0.0045	2.5640
	0.0025	0.0022	2.6410	0.0025	0.0024	2.6510	0.0025	0.0020	2.6250
	0.0010	0.0004	2.7010	0.0010	0.0004	2.7110	0.0010	0.0003	2.6820
	0.1000	0.1075	1.2470	0.1000	0.1014	1.2320	0.1000	0.1233	1.2200
	0.0500	0.0439	1.2940	0.0500	0.0497	1.5530	0.0500	0.0487	1.2680
	0.0250	0.0193	2.0790	0.0250	0.0273	1.9970	0.0250	0.0237	2.0340
7	0.0100	0.0079	2.1200	0.0100	0.0081	2.1060	0.0100	0.0109	2.0820
	0.0050	0.0040	2.1580	0.0050	0.0050	2.1420	0.0050	0.0047	2.1210
	0.0025	0.0028	2.8200	0.0025	0.0024	2.5700	0.0025	0.0028	2.7780
	0.0010	0.0015	2.9100	0.0010	0.0010	2.8760	0.0010	0.0015	2.8480
	0.1000	0.0966	1.2270	0.1000	0.0990	1.4980	0.1000	0.0980	1.2140
	0.0500	0.0602	1.5600	0.0500	0.0468	1.5850	0.0500	0.0587	1.5490
	0.0250	0.0260	1.6370	0.0250	0.0260	1.6180	0.0250	0.0251	1.8210
8	0.0100	0.0084	2.3410	0.0100	0.0122	2.3140	0.0100	0.0086	2.3240
	0.0050	0.0042	2.4090	0.0050	0.0064	2.3770	0.0050	0.0043	2.3870
	0.0025	0.0022	2.4540	0.0025	0.0026	2.4280	0.0025	0.0022	2.4290
	0.0010	0.0009	2.8280	0.0010	0.0011	2.9970	0.0010	0.0010	2.8280
	0.1000	0.0846	1.1640	0.1000	0.1038	1.1460	0.1000	0.0809	1.1510
	0.0500	0.0532	1.7830	0.0500	0.0503	1.5580	0.0500	0.0554	1.7730
	0.0250	0.0294	1.9010	0.0250	0.0199	1.8830	0.0250	0.0275	1.8830
9	0.0100	0.0090	1.9650	0.0100	0.0100	2.2230	0.0100	0.0094	1.9430
	0.0050	0.0061	2.5820	0.0050	0.0053	2.5040	0.0050	0.0061	2.5620
	0.0025	0.0014	2.7160	0.0025	0.0019	2.6740	0.0025	0.0033	2.6370
	0.0010	0.0014	2.7250	0.0010	0.0009	2.7260	0.0010	0.0013	2.6960
	0.1000	0.0989	1.3500	0.1000	0.1084	1.4080	0.1000	0.1010	1.3630
	0.0500	0.0439	1.5090	0.0500	0.0504	1.4850	0.0500	0.0438	1.4910
	0.0250	0.0246	1.8860	0.0250	0.0269	2.0380	0.0250	0.0253	1.8650
10	0.0100	0.0102	2.2160	0.0100	0.0085	2.2280	0.0100	0.0104	2.1920
	0.0050	0.0043	2.3040	0.0050	0.0044	2.2580	0.0050	0.0049	2.2970
	0.0025	0.0025	2.6420	0.0025	0.0030	2.8160	0.0025	0.0025	2.6250
	0.0010	0.0009	2.9550	0.0010	0.0007	2.9810	0.0010	0.0009	2.9230
	0.1000	0.0839	1.0990	0.1000	0.1051	1.0880	0.1000	0.1117	1.0740
	0.0500	0.0480	1.7280	0.0500	0.0540	1.6420	0.0500	0.0455	1.7050
	0.0250	0.0289	1.8130	0.0250	0.0223	1.8130	0.0250	0.0272	1.7900
11	0.0100	0.0110	2.3840	0.0100	0.0097	2.2020	0.0100	0.0114	2.3630
	0.0050	0.0046	2.5310	0.0050	0.0050	2.4830	0.0050	0.0043	2.4990
	0.0025	0.0021	2.5660	0.0025	0.0027	2.5400	0.0025	0.0021	2.5320
	0.0010	0.0010	3.0650	0.0010	0.0010	2.9030	0.0010	0.0011	3.0380
	0.1000	0.1022	1.3320	0.1000	0.1053	1.3600	0.1000	0.0955	1.3380
	0.0500	0.0471	1.4200	0.0500	0.0478	1.4140	0.0500	0.0490	1.4980
	0.0250	0.0262	1.9080	0.0250	0.0262	2.0410	0.0250	0.0275	1.9400
12	0.0100	0.0106	2.1290	0.0100	0.0087	2.1210	0.0100	0.0104	2.1010
	0.0050	0.0051	2.4820	0.0050	0.0052	2.6220	0.0050	0.0049	2.4520
	0.0025	0.0022	2.7410	0.0025	0.0021	2.7900	0.0025	0.0029	2.6770
	0.0010	0.0012	2.8380	0.0010	0.0009	2.8280	0.0010	0.0010	2.8020
	0.1000	0.1035	1.0400	0.1000	0.0999	1.2720	0.1000	0.1041	1.0250
	0.0500	0.0454	1.7110	0.0500	0.0559	1.6350	0.0500	0.0441	1.6860
	0.0250	0.0266	1.7340	0.0250	0.0248	1.7350	0.0250	0.0268	1.7090
13	0.0100	0.0089	2.3950	0.0100	0.0109	2.2890	0.0100	0.0089	2.3600
	0.0050	0.0041	2.4280	0.0050	0.0050	2.4040	0.0050	0.0046	2.3930
	0.0025	0.0018	2.6180	0.0025	0.0027	2.6980	0.0025	0.0026	2.8470
	0.0010	0.0009	3.0790	0.0010	0.0009	3.0350	0.0010	0.0010	3.0430
	0.1000	0.0946	1.3190	0.1000	0.0960	1.3260	0.1000	0.0934	1.2990
	0.0500	0.0503	1.6580	0.0500	0.0451	1.6030	0.0500	0.0517	1.6340
	0.0250	0.0236	1.9780	0.0250	0.0255	1.9890	0.0250	0.0238	1.9480
14	0.0100	0.0097	2.3080	0.0100	0.0081	2.2460	0.0100	0.0101	2.2870
	0.0050	0.0053	2.4610	0.0050	0.0042	2.6530	0.0050	0.0053	2.5330
	0.0025	0.0022	2.7040	0.0025	0.0019	2.6900	0.0025	0.0024	2.6260
	0.0010	0.0010	2.9840	0.0010	0.0008	2.8070	0.0010	0.0010	2.9950

Table C.8: Critical points, $\hat{T}_\alpha\{3 \ldots 14, 23 \ldots 25\}$, for the $\hat{T}$ distribution.

m, n	23			24			25		
	0.1000	0.0924	1.2900	0.1000	0.1032	1.2870	0.1000	0.0942	1.2900
	0.0500	0.0549	1.6390	0.0500	0.0554	1.6090	0.0500	0.0542	1.6130
	0.0250	0.0223	1.9090	0.0250	0.0266	1.9310	0.0250	0.0240	1.8070
15	0.0100	0.0122	2.2940	0.0100	0.0113	2.2530	0.0100	0.0124	2.2590
	0.0050	0.0034	2.4540	0.0050	0.0049	2.5610	0.0050	0.0037	2.4440
	0.0025	0.0029	2.8570	0.0025	0.0026	2.6850	0.0025	0.0029	2.8170
	0.0010	0.0008	2.9860	0.0010	0.0010	2.9540	0.0010	0.0010	2.9390
	0.1000	0.1071	1.2720	0.1000	0.1045	1.2750	0.1000	0.1071	1.2520
	0.0500	0.0519	1.6220	0.0500	0.0503	1.7680	0.0500	0.0537	1.5960
	0.0250	0.0232	1.9470	0.0250	0.0196	1.9360	0.0250	0.0237	1.9150
16	0.0100	0.0113	2.2720	0.0100	0.0096	2.4370	0.0100	0.0104	2.2350
	0.0050	0.0042	2.5960	0.0050	0.0063	2.5500	0.0050	0.0057	2.5040
	0.0025	0.0022	2.6380	0.0025	0.0035	2.5810	0.0025	0.0026	2.7230
	0.0010	0.0010	2.9310	0.0010	0.0012	3.0860	0.0010	0.0011	3.0250
	0.1000	0.0992	1.1640	0.1000	0.1035	1.2640	0.1000	0.1000	1.4340
	0.0500	0.0419	1.5990	0.0500	0.0530	1.5800	0.0500	0.0435	1.5710
	0.0250	0.0255	2.0340	0.0250	0.0288	1.8960	0.0250	0.0244	2.0770
17	0.0100	0.0098	2.2380	0.0100	0.0123	2.2120	0.0100	0.0096	2.2000
	0.0050	0.0044	2.3220	0.0050	0.0055	2.5280	0.0050	0.0050	2.5810
	0.0025	0.0028	2.8420	0.0025	0.0025	2.7870	0.0025	0.0030	2.7940
	0.0010	0.0015	2.8780	0.0010	0.0009	2.9370	0.0010	0.0007	2.9900
	0.1000	0.1023	1.2550	0.1000	0.0854	1.2470	0.1000	0.1014	1.2320
	0.0500	0.0437	1.5940	0.0500	0.0516	1.7840	0.0500	0.0467	1.5800
	0.0250	0.0285	1.8820	0.0250	0.0241	1.8700	0.0250	0.0204	1.8690
18	0.0100	0.0087	2.2500	0.0100	0.0095	2.4590	0.0100	0.0100	2.3280
	0.0050	0.0054	2.5100	0.0050	0.0048	2.4940	0.0050	0.0052	2.4660
	0.0025	0.0026	2.7640	0.0025	0.0021	2.6700	0.0025	0.0026	2.7730
	0.0010	0.0010	2.9560	0.0010	0.0012	3.0780	0.0010	0.0012	3.0190
	0.1000	0.0983	1.4660	0.1000	0.1113	1.2240	0.1000	0.0999	1.4590
	0.0500	0.0496	1.5500	0.0500	0.0544	1.5310	0.0500	0.0500	1.5210
	0.0250	0.0261	2.0650	0.0250	0.0230	1.8670	0.0250	0.0222	2.1080
19	0.0100	0.0118	2.1700	0.0100	0.0093	2.1940	0.0100	0.0126	2.1300
	0.0050	0.0050	2.6390	0.0050	0.0044	2.4680	0.0050	0.0046	2.7110
	0.0025	0.0020	2.7900	0.0025	0.0025	2.7550	0.0025	0.0022	2.7390
	0.0010	0.0008	3.0220	0.0010	0.0011	3.0610	0.0010	0.0010	2.8130
	0.1000	0.1115	1.2190	0.1000	0.0946	1.2110	0.1000	0.0870	1.2020
	0.0500	0.0467	1.5310	0.0500	0.0479	1.7980	0.0500	0.0492	1.6710
	0.0250	0.0217	1.8680	0.0250	0.0280	1.8160	0.0250	0.0243	1.8250
20	0.0100	0.0100	2.2750	0.0100	0.0098	2.3980	0.0100	0.0100	2.3360
	0.0050	0.0038	2.4430	0.0050	0.0062	2.4220	0.0050	0.0044	2.4100
	0.0025	0.0028	2.7430	0.0025	0.0026	2.9170	0.0025	0.0020	2.7860
	0.0010	0.0012	3.0480	0.0010	0.0011	3.0270	0.0010	0.0012	2.9910
	0.1000	0.0972	1.4520	0.1000	0.0892	1.2460	0.1000	0.1039	1.4260
	0.0500	0.0539	1.5090	0.0500	0.0469	1.4960	0.0500	0.0565	1.4800
	0.0250	0.0250	2.0910	0.0250	0.0252	1.8560	0.0250	0.0231	2.0510
21	0.0100	0.0073	2.2340	0.0100	0.0103	2.3270	0.0100	0.0085	2.2130
	0.0050	0.0054	2.6880	0.0050	0.0049	2.3870	0.0050	0.0050	2.6380
	0.0025	0.0027	2.7160	0.0025	0.0032	2.6810	0.0025	0.0030	2.6640
	0.0010	0.0009	2.9010	0.0010	0.0011	2.9790	0.0010	0.0010	3.1370
	0.1000	0.0874	1.2190	0.1000	0.1004	1.1800	0.1000	0.0920	1.1840
	0.0500	0.0494	1.5710	0.0500	0.0538	1.7520	0.0500	0.0518	1.6420
	0.0250	0.0250	1.9460	0.0250	0.0190	1.9410	0.0250	0.0246	1.9500
22	0.0100	0.0099	2.3250	0.0100	0.0073	2.3610	0.0100	0.0093	2.2980
	0.0050	0.0049	2.4640	0.0050	0.0038	2.5780	0.0050	0.0049	2.5650
	0.0025	0.0020	2.7800	0.0025	0.0022	2.9220	0.0025	0.0024	2.7200
	0.0010	0.0013	2.9730	0.0010	0.0006	3.2460	0.0010	0.0009	2.9250
	0.1000	0.0921	1.4590	0.1000	0.0945	1.1930	0.1000	0.1054	1.3900
	0.0500	0.0590	1.4740	0.0500	0.0501	1.6350	0.0500	0.0411	1.5580
	0.0250	0.0282	2.0430	0.0250	0.0255	1.9890	0.0250	0.0254	2.0020
23	0.0100	0.0085	2.1590	0.0100	0.0098	2.3060	0.0100	0.0100	2.3830
	0.0050	0.0063	2.6270	0.0050	0.0054	2.5590	0.0050	0.0059	2.5740
	0.0025	0.0034	2.6530	0.0025	0.0024	2.7200	0.0025	0.0018	2.7750
	0.0010	0.0010	3.2110	0.0010	0.0009	2.9920	0.0010	0.0010	3.1460
				0.1000	0.1044	1.1540	0.1000	0.0980	1.1460
				0.0500	0.0570	1.7140	0.0500	0.0515	1.6810
				0.0250	0.0214	1.8790	0.0250	0.0227	1.9660
24				0.0100	0.0089	2.3090	0.0100	0.0106	2.2810
				0.0050	0.0046	2.5060	0.0050	0.0048	2.5660
				0.0025	0.0018	2.8860	0.0025	0.0025	2.8020
				0.0010	0.0006	3.0070	0.0010	0.0011	3.0820
							0.1000	0.1002	1.3990
							0.0500	0.0451	1.5030
							0.0250	0.0269	1.9600
25							0.0100	0.0107	2.4470
							0.0050	0.0048	2.5450
							0.0025	0.0022	2.7010
							0.0010	0.0008	3.1110

Table C.9: Critical points, $\hat{T}_\alpha\{15\ldots25, 23\ldots25\}$, for the $\hat{T}$ distribution.